TSGym: Design Choices for Deep Multivariate Time-Series Forecasting

Shuang Liang[1,*], **Chaochuan Hou**[1,*], **Xu Yao**[1], **Shiping Wang**[3],
Minqi Jiang[1,*,†], **Songqiao Han**[1,2,†], **Hailiang Huang**[1,2,†],
[1]AI Lab, Shanghai University of Finance and Economics [2]Ant Group
[3]MoE Key Laboratory of Interdisciplinary Research of Computation and Economics
{liangs1104,houchaochuan,yaoxu}@stu.sufe.edu.cn, shiping.wsp@antgroup.com,
{jiangminqi,han.songqiao,hlhuang}@shufe.edu.cn,

Abstract

Recently, deep learning has driven significant advancements in multivariate time series forecasting (MTSF) tasks. However, much of the current research in MTSF tends to evaluate models from a holistic perspective, which obscures the individual contributions and leaves critical issues unaddressed. Adhering to the current modeling paradigms, this work bridges these gaps by systematically decomposing deep MTSF methods into their core, fine-grained components like series-patching tokenization, channel-independent strategy, attention modules, or even Large Language Models and Time-series Foundation Models. Through extensive experiments and component-level analysis, our work offers more profound insights than previous benchmarks that typically discuss models as a whole.

Furthermore, we propose a novel automated solution called TSGym for MTSF tasks. Unlike traditional hyperparameter tuning, neural architecture searching or fixed model selection, TSGym performs fine-grained component selection and automated model construction, which enables the creation of more effective solutions tailored to diverse time series data, therefore enhancing model transferability across different data sources and robustness against distribution shifts. Extensive experiments indicate that TSGym significantly outperforms existing state-of-the-art MTSF **and AutoML** methods. All code is publicly available on https://github.com/SUFE-AILAB/TSGym.

1 Introduction

Multivariate time series refer to time series data involving multiple interdependent variables, which are widely present in various fields such as finance [52], energy [6, 16], traffic [14, 70], and health [9, 30]. Among the numerous analysis tasks, multivariate time series forecasting (MTSF) attracts substantial attention from the research community due to its significant practical applications. Traditional approaches to MTSF are largely based on statistical methods [4, 73] and machine learning techniques [23, 44]. In recent years, deep learning (DL) has become the most active area of research for MTSF, driven by its ability to handle complex patterns and large-scale datasets effectively [60].

Early academic efforts of deep MTSF methods like RNN-type methods [68] are reported to struggle with capturing long-term temporal dependencies due to their inherent limitations of gradient vanishing or exploding problem [76, 78]. More recently, Transformer [57] shows significant potential, largely due to the effectiveness of its attention mechanisms in modeling temporal correlation [57, 63]. Consequently, attention mechanism has continuously been studied in MTSF, with a focus on adapting them to time series data, for instance, by exploiting sparsity inductive bias [33, 76], transforming time and frequency domains [78], and fusing multi-scale series [37]. While simpler MLP-based structures

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

emerged [71] offering alternatives to the established Transformer architecture in MTSF, notable modeling strategies like series-patching and channel-independent [46], significantly enhanced the performance of Transformer-based methods, thereby sustaining research interest in them. Building upon these developments, large time-series models including large language models (LLMs) [29, 79, 26] and time series foundation models (TSFMs) [28, 43] have recently been introduced, achieving promising results and fostering new research directions for MTSF. Alongside these advancements in model architectures, active research within the deep MTSF community also focuses on other critical topics, such as variable (channels) dependency modeling [46, 38, 75], series normalization methods [41, 17], and trend-seasonal decomposition [71, 39].

As the field of MTSF continues to diversify, existing studies typically address critical concerns about methodological effectiveness, either by conducting large-scale benchmarks [60, 53, 49] or performing model selection via AutoML [2, 19]. However, we identify three main challenges with these prevailing approaches: First, *the granularity of existing studies is insufficient*. Current benchmarking works evaluate or select models as a whole, which hinders a deeper understanding of the mechanisms that drive model performance. In AutoML, this lack of granularity prevents breakthroughs beyond the limits of existing models. Second, *the scope of existing studies is limited*. Current benchmarking and automated selection efforts are often confined to restricted model architectures or hyperparameters, without covering a broad range of data processing methods or feature modeling techniques. Third, *the range of existing studies is narrow*. Existing studies tend to cover only a subset of network architectures and often lack discussions on more diverse models, such as LLMs and TSFMs.

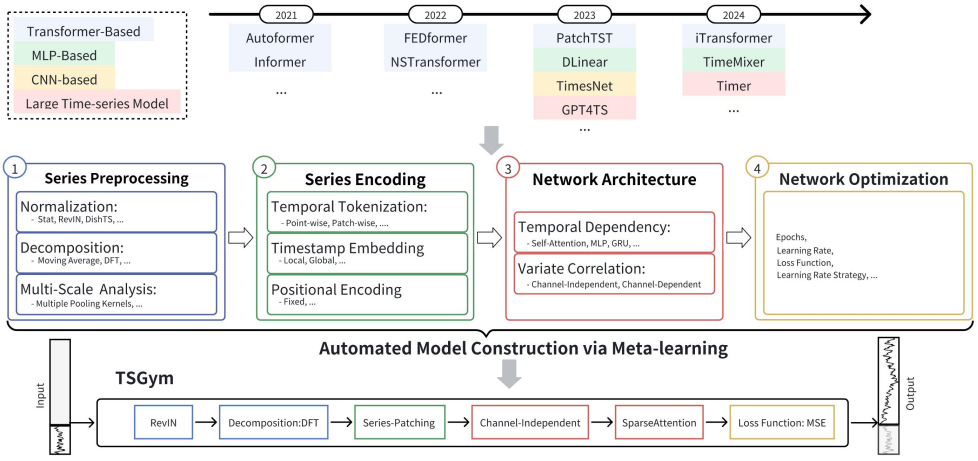


Figure 1: The Pipeline of Existing Multivariate Time Series Forecasting Methods.

To bridge these gaps, we propose TSGym—a framework designed for the **Large-scale Evaluation**, **Analysis**, and **Automated Model Construction** in deep MTSF tasks. Rather than viewing models as unified entities, TSGym systematically deconstructs popular deep MTSF methods by organizing them into distinct design dimensions that cover the entire time series modeling pipeline (see Fig. 1 and Table 1). Through extensive experiments, TSGym conducts fine-grained, isolated evaluations of core components, thereby identifying key design dimensions/choices and valuable insights from the vast MTSF methods. **Leveraging a large-scale benchmark, TSGym systematically validates or refutes a series of prevailing claims within the MTSF community, including comparisons between Transformer and MLP architectures, as well as the adaptability of channel-independent approaches.** Moreover, TSGym proposes the first component-level model construction in MTSF tasks, which effectively overcomes limitations in the previous automation methods by enabling more flexible and customized model designs tailored to data characteristics. Extensive experimental results indicate that the proposed TSGym generally outperforms existing SOTA methods. We summarize the key contributions of TSGym as follows:

Component-level evaluation of MTSF methods. We propose TSGym, the first large-scale benchmark that systematically decouples deep MTSF methods. By evaluating 16 design dimensions across 10 benchmark datasets, TSGym **elucidates contested issues in the current community** and offers key insights to inform future development for MTSF.

Automated MTSF model construction. Leveraging meta-learning, TSGym develops models that outperform current SOTA methods, offering the MTSF community an effective, automated, and data-adaptive solution for model design.

Discussion on emerging large time-series models. TSGym broadens current MTSF scope by applying systematic evaluation and automated combination not only to well-established models like MLP and Transformer, but also to novel large time-series models like LLMs and TSFMs.

2 Related Work

2.1 Deep Learning-based MTSF

MTSF evolves from traditional statistical methods like ARIMA and Gaussian processes to modern deep learning approaches. Recurrent Neural Networks (RNNs) introduce memory mechanisms for sequential data but struggle with long-term dependencies. Temporal Convolutional Networks (TCNs) improve this by capturing multi-scale patterns, though their fixed window sizes limit global context. Transformers, using self-attention, enable long-range forecasting but introduce high computational complexity, leading to efficient variants like sparse attention [66] and patch-based models [46]. Multilayer Perceptrons (MLPs) regain attention as simple yet effective models [72], with numerous variants offering competitive performance [13, 69, 15, 39]. Leveraging NLP foundation models, LLM adaptation approaches use frozen backbones and prompt engineering [27, 79] or fine-tuning [11] to transfer pretrained knowledge. Simultaneously, pure TSFMs trained on large datasets achieve zero-shot generalization [43, 20], though constrained by Transformers' complexity. Our TSGym framework modularizes six core backbones—RNNs, CNNs, Transformers, MLPs, LLMs, and TSFMs—offering flexible, hybrid integration based on temporal dependencies and resource needs.

In recent advancements in MTSF, we summarize the design paradigm through a unified pipeline (Fig. 1), consisting of four stages: *Series Preprocessing→Series Encoding→Network Architecture→Network Optimization*. Additionally, several specialized modules are proposed to enhance predictive accuracy by addressing non-stationarity, multi-scale dependencies, and inter-variable interactions. We categorize these developments into 6 specialized modules:

(1) Normalization methods like RevIN [31] adjust non-stationary data, improving robustness against distribution shifts. **(2) Decomposition methods**, such as Autoformer [66]'s trend-seasonality separation, isolate non-stationary components, making the data more predictable by separating trends from seasonality. **(3) Multi-scale analysis** extracts temporal patterns across granularities, as in TimeMixer [59], capturing both high-frequency fluctuations and low-frequency trends through hierarchical resolution modeling. **(4) Temporal tokenization techniques** like PatchTST[46]'s subseries-level embedding represent time series hierarchically, improving the capture of complex temporal semantics. **(5) Temporal dependency** modeling through architectures like Transformers leverages self-attention to capture long-range dependencies, effectively modeling both short- and long-term relationships. **(6) Variate correlation learning**, exemplified by DUET [50], models inter-variable dependencies using frequency-domain metric learning, improving predictions by capturing interactions across variables.

To provide a more detailed categorization and comprehensive technical specifications, please refer to Appx. B. Due to the extensive focus and continuous evolution of these modules in MTSF research, TSGym strives to decouple and modularize these key modules, exploring their real contributions and enabling more flexible model structure selection and configuration.

2.2 Benchmarks for Time Series Forecasting

Recent time series forecasting benchmark studies [60, 53, 49, 42] have conducted large-scale experiments across a diverse range of datasets. However, most of these works treat current models as monolithic entities. TSlib[1] [60], one of the most popular repositories for time series analysis, provides a comprehensive survey and evaluates recent time series models across various time series analysis tasks. From the perspective of time series characteristics, BasicTS [53] analyzes model architectures and the strategy of treating channels (or variables) independently. With a more extensive experimental setup, TFB [49] additionally includes machine learning and statistical forecasting methods, and covers datasets from a broader range of domains. More recently, OpenLTM[2] [42] provides a

[1] https://github.com/thuml/Time-Series-Library

[2] https://github.com/thuml/OpenLTM

system to evaluate Time Series Foundation Models as well as Large language Models for time series methods. Although some surveys and benchmarks analyze the fine-grained components of time series models, their scope is often limited. Wen et al. [62] discuss various time series data augmentation techniques and evaluate their effectiveness. Another survey [63] systematically reviews fine-grained components within Transformer-based architectures, but lacks broader coverage of model structures and experimental evaluations.

These limitations prevent the aforementioned studies from comprehensively and meticulously evaluating the entire MTSF pipeline, spanning from sequence preprocessing to model parameter optimization. To the best of our knowledge, TSGym is the first benchmark that not only provides component-level fine-grained analysis, but also conducts large-scale empirical evaluations.

2.3 AutoML for Time Series Forecasting

Current automated approaches for DL-based MTSF can be categorized into ensemble-based [54] and meta-learning-based [2, 19] methods. The former fits and integrates various models from a predefined pool with ensemble techniques, which inevitably incurs substantial computational cost. The latter leverages meta-features to characterize datasets and selects optimal models for the given datasets. However, both approaches operate at the model level and struggle to surpass the performance ceiling of existing methods. AutoCTS++ [67] achieves automated selection by searching over model architectures and hyperparameters, but its search space is limited in scope. In contrast, TSGym is the first framework to support automated selection over a wide range of fine-grained components for MTSF, extending beyond narrow model structures, hyperparameters, and data processing strategies.

A closely related work is our previous effort, ADGym[24], which is designed for tabular anomaly detection with model decomposition. Differently, TSGym deals with multivariate time series data, which presents more complex data processing design choices, such as series sampling, series normalization, and series decomposition. **Second,** TSGym considers finer-grained model structures, such as various attention variants in Transformers, and broader network types, including LLMs and TSFMs. **Third, TSGym explores the value of Optuna[5], a Bayesian-optimization-driven intelligent search framework, which attains a superior design space at markedly lower cost and thus enhances the efficacy of TSGym.** It is worth mentioning that the success of TSGym validates the universality of the model decomposition framework, marking an innovation and progression distinct from ADGym. Further details on the differences between two works can be found in Appx.F.

3 TSGym: *Benchmarking* and *Automating* Design Choices in Deep MTSF

3.1 Problem Definition for MTSF

In this paper, we focus on the common MTSF settings for time series data containing C variates. Given historical data $\chi = \{\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_C^t\}_{t=1}^{L}$, where L is the look-back sequence length and $\boldsymbol{x}_i^t$ is the i-th variate, the forecasting task is to predict T-step future sequence $\hat{\chi} = \{\hat{\boldsymbol{x}}_1^t, \ldots, \boldsymbol{x}_C^t\}_{t=L+1}^{L+T}$. To avoid error accumulation ($T > 1$), we directly predict all future steps, following [76].

3.2 Large Benchmarking towards Design Choices of Deep MTSF

Considering the above numerous methods **proposed** for MTSF tasks, the foremost priority involves decoupling the current state-of-the-art (SOTA) methods and further conducting large-scale benchmark to identify the core components that really drive the improvements in time-series forecasting.

Following the taxonomy of the previous study [63, 72], we decouple existing SOTA methods according to the standard process of MTSF modeling, while significantly expanding the diversity of the modeling pipeline. Based on the flow direction from the input to the output sequence, the **Pipeline** of TSGym includes: *Series Preprocessing→Series Encoding→Network Architecture→Network Optimization*, as is demonstrated in Fig. 1. Moreover, we structure each pipeline step according to distinct **Design Dimensions**, where a DL-based time-series forecasting model can be instantiated by specified **Design Choices**, as is shown in Table 1.

Through the proposed design dimensions and choices, TSGym provides detailed description of time-series modeling pipeline, disentangling key elements within mainstream time-series forecasting methods and facilitating component-level comparison/automated construction. For example, TSGym includes multi-scale mixing module proposed in TimeMixer [59], Inverted Encoding method proposed in iTransformer [38], Channel-independent strategy and Series-Patching encoding used in

Table 1: TSGym supports comprehensive design choices for deep time-series forecasting methods.

Pipeline	Design Dimensions	Design Choices
↓Series Preprocessing	Series Normalization	[None, Stat, RevIN, DishTS]
	Series Decomposition	[None, MA, MoEMA, DFT]
	Series Sampling/Mixing	[False, True]
↓Series Encoding	Channel Independent	[False, True]
	Sequence Length	[48, 96, 192, 512]
	Series Embedding	[Inverted Encoding, Positional Encoding, Series Patching]
↓Network Architecture	Network Type	[MLP, RNN, Transformer, LLM, TSFM]
	Series Attention	[Null, SelfAttn, AutoCorr, SparseAttn, FrequencyAttn, DestationaryAttn]
	Feature Attention	[Null, SelfAttn, SparseAttn, FrequencyAttn]
	d_model	[64, 256]
	d_ff	[256, 1024]
	Encoder Layers	[2, 3]
↓Network Optimization	Epochs	[10, 20, 50]
	Loss Function	[MSE, MAE, HUBER]
	Learning Rate	[1e-3, 1e-4]
	Learning Rate Strategy	[Null, Type1]

PatchTST [46], various attention mechanism discussed in [63], and also LLM and TSFM network type choices that are often integrated without fully considering their interactions with other design dimensions. Detailed descriptions of all design choices are provided in Appx. G.1.

3.3 Automated construction MTSF models via TSGym

Overview. Differing from traditional methods that focus on selecting an off-the-shelf model, TSGym aims to customize models given the downstream MTSF tasks and data descriptions. Given a pre-defined conflict-free model set $\mathcal{M} = \{M_1, ..., M_m\}$, each model M_i is instantiated by the design choice combinations illustrated in Table 1. TSGym learns the mapping function from these automatically combined models to their associated forecasting performance on the training datasets, and generalize to the test dataset(s) to select the best model based on predicted results.

Meta-learning for automated MTSF model construction. Formally speaking, TSGym propose k design dimensions $\mathcal{DD} = \{DD_1, ..., DD_k\}$ for comprehensively describing each step of aforementioned pipeline in deep learning time-series modeling. Each design dimension DD_i represents a set containing elements of different design choices DC. By taking the Cartesian product of the sets $\mathcal{DD}$ corresponding to different design dimensions, we obtain the pool of all valid model combinations $\mathcal{M} = DD_1 \times DD_2 \times \cdots \times DD_k = \{(DC_1, DC_2, \ldots, DC_k) \mid DC_i \in DD_i, i = 1, 2, \ldots, k\}$. Considering the potentially large number of combinations and the computational cost, we randomly sampled $\mathcal{M}$ to $\mathcal{M}_s$, where $M_i = (DC_1 = RevIN, DC_1 = DFT, ..., DC_k = Type1) \in \mathcal{M}_s$, for example, which means M_i instantiates RevIN method to normalize input series, then decompose it to the seasonal and trend term. Subsequently, following the Series Encoding and Network Architecture constructing pipeline (as illustrated in Table 1), finally the Type1, i.e., a step decay learning rate strategy is employed to adjust the learning rate for updating the model parameters.

Suppose we have n training datasets $\boldsymbol{\mathcal{D}}_{\text{train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ and the number of sampled model combinations (i.e., the size of the set $\mathcal{M}_s$) is m, TSGym conducts extensive experiments on n historical training datasets to evaluate and further collect the forecasting performance of m model combinations. TSGym then acquire the MSE performance matrix $\boldsymbol{P} \in \mathbb{R}^{n \times m}$, where $\boldsymbol{P}_{i,j}$ corresponds to the j-th auto-constructed MTSF model's performance on the i-th training dataset. Since the difficulty of prediction tasks varies across training datasets, leading to significant differences in the numerical range of performance metrics. Directly using these metrics (e.g., MSE) as training targets of a meta-predictor may result in overfitting on more difficult dataset(s). Therefore, we convert the performance metrics of $\mathcal{M}_s$ into their corresponding normalized ranking, where $\boldsymbol{R}_{i,j} = rank(P_{i,j})/m \in [0, 1]$ and smaller values indicate better performance on the corresponding dataset.

Distinguished from previous model selection approaches [2, 3], TSGym decouples more recently MTSF methods (including MLP-Mixer-type, Transformer-based, LLM and TSFM models), and supports fine-grained model construction at the component level, rather than being constrained to a fixed, limited set of existing models, which enables significantly greater flexibility and effectiveness. Specifically, TSGym follows the idea of meta-learning to construct a meta-predictor that learns the mapping function $f(\cdot)$ from training dataset $\mathcal{D}_i$ and model combination M_j, to the performance rankings $\boldsymbol{R}_{i,j}$, as is shown in Eq. 1. We leverage meta-features $\mathbf{E}_i^{meta}$ **from the training set of each**

dataset, which capture multiple aspects such as statistical, temporal, spectral, fractal features, and distribution shift metrics to fully describe the complex data characteristics of time series datasets. Learnable continuous embeddings $\mathbf{E}_j^{comp}$ are used to represent different model combinations and are updated through the gradient backpropagation of the meta-predictor. **At test time, only the meta-features extracted from the training split of the target dataset are required to identify the top-performing component, eliminating any redundant experimentation on the test set.**

$$f(\mathcal{D}_i, M_j) = \boldsymbol{R}_{i,j}, f \ : \ \underbrace{\mathbf{E}_i^{meta}}_{\text{meta features}} \ , \ \underbrace{\mathbf{E}_j^{comp}}_{\text{component embed.}} \ \mapsto \boldsymbol{R}_{i,j} \ , \ i \in \{1, \ldots, n\}, \ j \in \{1, \ldots, m\} \quad (1)$$

We used a simple two-layer MLP as the meta-predictor and trained it through a regression problem, thereby transferring the learned mapping to new test datasets. For a newcoming dataset (i.e., test dataset $\mathbf{X}_{\text{test}}$), we acquire the predicted relative ranking of different components using the trained $f(\cdot)$, and select top-1 (k) to construct MTSF model(s). Note this procedure is zero-shot without needing any neural network training on $\mathbf{X}_{\text{test}}$ but only extracting meta-features and pipeline embeddings. We show the effectiveness of the meta-predictor in §4.3.

4 Experiments

4.1 Experiment Settings

Datasets. Following most prior works [66, 65, 27], we adopt 9 datasets as experimental data for MTSF tasks, ETT (4 subsets), Traffic, Electricity, Weather, Exchange, ILI. And we utilize the M4 dataset for short-term forecasting tasks. The forecast horizon L for long-term forecasting is $\{96, 192, 336, 720\}$, while for the ILI dataset, it is $\{24, 36, 48, 60\}$. For short-term forecasting, the forecast horizons are $\{6, 8, 13, 14, 18, 48\}$. More details can be seen in Appx. A.

Baseline. We present a comprehensive set of baseline comparison experiments **including MTSF and AutoML methods,** to demonstrate the superior performance of the pipelines automatically constructed by TSGym. Due to space limitations, the baseline methods presented in this section include the latest clustering-based approach DUET [50], time series mixing methods (TimeMixer [59]), MLP-based methods (MICN [58]), RNN-based methods (SegRNN [35]), CNN-based methods (TimesNet [65]), and Transformer-based methods (PatchTST [46], Crossformer [75], Autoformer [66]). We present experiments based on the complete baseline in the Appx. H.

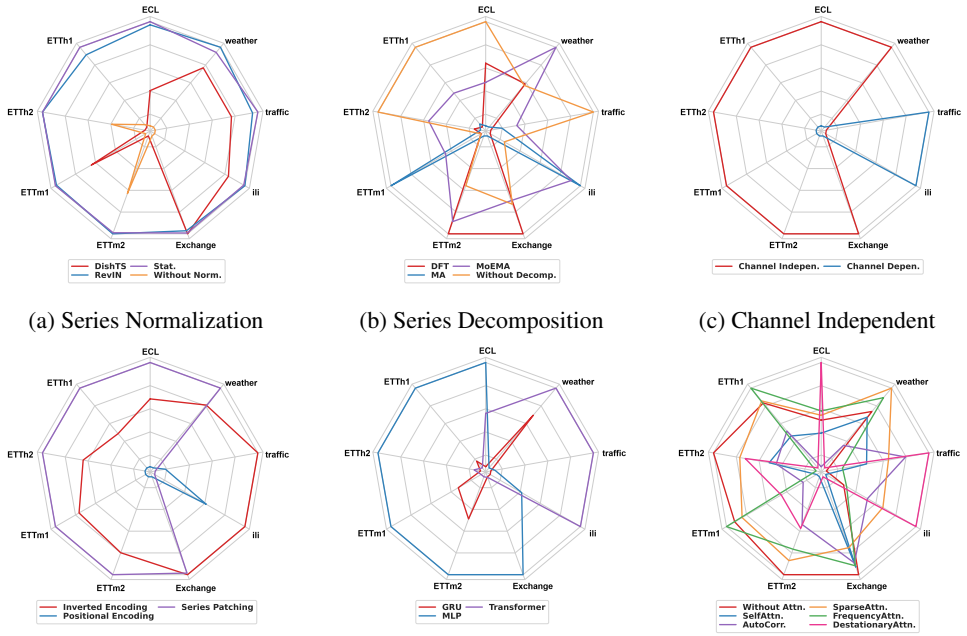
Evaluation Metrics. We follow the experimental setup of most prior works, using Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics for long-term forecasting tasks, and using Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Scaled Error (MASE), and Overall Weighted Average (OWA) as metrics for short-term forecasting tasks. The mathematical formulas for these evaluation metrics are provided in the Appx. D.

Meta-predictor in TSGym. The meta-predictor is instantiated as a two-layer MLP and trained for 100 epochs with early stopping. The training process utilizes the Adam optimizer [32] with a learning rate of 0.001 and batch size of 512. See details in Appx. G.2.

4.2 Large Evaluation on Time-Series Design Choices

In this work, we perform large evaluations on the decoupled pipelines according to the standard procedure of MTSF methods. Such analysis is often overlooked in previous studies, and we investigate each design dimension of decoupled pipelines by fixing its corresponding design choice (e.g., Self Attention), and randomly sampling other dimensional design choices to construct MTSF pipelines.

In the following sections, we provide systematic conclusions based on long-term MTSF experimental results, addressing several gaps in the current MTSF research community. Specifically, different design choices are compared and demonstrated using a spider chart, where each vertex represents a dataset. Design choices that are closer to the vertices exhibit superior performance in their respective design dimensions. We analyze these components based on the different design dimensions, namely ***Series Preprocessing***, ***Series Encoding*** and ***Network Construction***. Finally, we evaluated the performance on four datasets under the fixed model architectures of LLMs or TSFMs, and discussed the effects of various design choices in the subsection on ***Large Time Series Models***. **Figure 2 presents the average results of key components across diverse datasets and forecast horizons.** More detailed information, including the complete design choices and their performance on short-term



(a) Series Normalization (b) Series Decomposition (c) Channel Independent

(d) Series Tokenization (e) Network Backbone (f) Series Attention

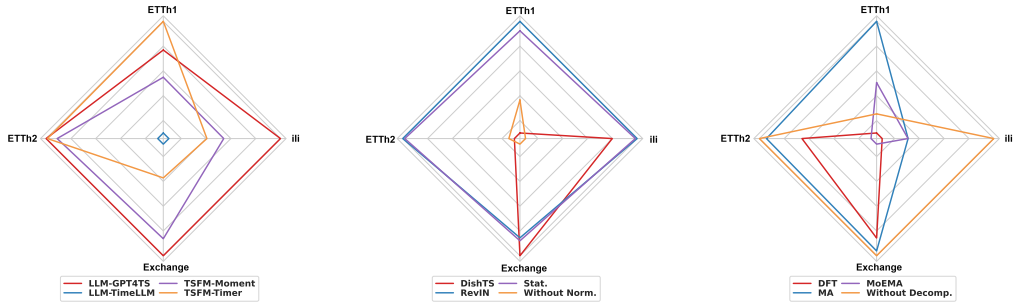
Figure 2: Overall performance across key design dimensions. The results (MSE) are based on the top 25th percentile across all forecasting horizons.

time series forecasting tasks, is provided in the Appx. H. **Based on the comprehensive experiments results, we conduct an in-depth investigation of the components, formulate the seven most contentious claims in the research community, and clarify them with our benchmark; the results are presented in Appx. K.**

Series Preprocessing. Based on the systematic evaluation of Series Preprocessing strategies in Fig. 2a and Fig. 2b, several key insights emerge. Series Normalization (Fig. 2a) proves universally effective, with RevIN and Stationary achieving the lowest MSE (top 25th percentile) across diverse datasets, establishing them as essential baselines for stabilizing non-stationary dynamics. In contrast, Series Decomposition (Fig. 2b) demonstrates varying effectiveness depending on the dataset. For example, decomposition methods improve performance on datasets like ETTm1 when using MA, while others, such as ETTh1, ECL, ETTh2, and traffic datasets, perform better without decomposition.

Series Encoding. Several clear observations emerge from the Series Encoding stage, as illustrated in Fig.2c and Fig.2d. First, as shown in panel Fig. 2c, channel-independent methods consistently outperform channel-dependent ones, with the exception of traffic and ILI, highlighting the advantage of modeling each variable separately. Regarding tokenization strategies Fig. 2d, both patch-wise (sequence patching) and series-wise (inverted encoding) approaches outperform the traditional point-wise encoding. Among them, patch-wise encoding shows strong performance across most datasets, whereas inverted encoding proves particularly effective on the traffic dataset.

Network Construction. As is shown in Fig. 2e, we find that complex network architectures like the Transformer are not always necessary, which only performs better than MLP on weather, traffic and ILI datasets. This result is reasonable since more sophisticated DL backbone like Transformer and its variants often require specific hyperparameter combinations and tailored architectural designs to achieve satisfactory performance. We indicate that these observed results further emphasize the importance of automated model construction tailored to specific data characteristics. Indeed, we show in § 4.3 that including the Transformer architecture in the set of design choices would enhance the performance of constructed model via TSGym. Fig. 2f suggests that employing Attention mechanisms for modeling dependencies of input sequence does not offer significant advantages, which aligns with recent findings [71]. Additionally, the spider chart shows no significant performance differences were observed among different variants of attention mechanisms.



(a) Network Backbone (b) Series Normalization (c) Series Decomposition

Figure 3: Overall performance of different design choices across 3 design dimensions (Network Backbone 3a, Series Normalization 2a, Series Decomposition 2b) when using LLMs or TSFMs. The results (MSE) are based on the top 25th percentile across all forecasting horizons.

Large Time Series Models. We choose three design dimensions crucial to large time-series models for discussion, as is shown in Fig.3. The most surprising finding pertains to the model backbones. We observe that GPT4TS demonstrates stable and competitive performance across most datasets. In contrast, Time-LLM, which is also based on LLMs, exhibits opposite results. A highly plausible explanation is that time series embeddings processed through different methods struggle to align consistently with the embedding space of word representations, which is an important part in Time-LLM, resulting in suboptimal performance of Time-LLM under diverse experimental configurations.

4.3 Automatic Component Construction via TSGym

Extensive experimental results discussed above indicate that in deep time series modeling, most design choices are determined by data characteristics, meaning one-size-fits-all approaches are seldom effective. This, in turn, emphasizes the necessity of automated model construction.

In this subsection, we compare the MTSF pipeline selected by TSGym with existing SOTA methods. Through large-scale experiments, we found that TSGym outperforms existing SOTA models in both long- and short-term MTSF tasks. Regarding algorithm efficiency, our experiments demonstrate that even when limited to a search pool of lightweight model structures, such as MLP and RNNs, TSGym can still achieve competitive results. We analyze the effectiveness of the pipelines automatically constructed by TSGym through five key questions as follows. Additional details, such as the results based on more metrics and more complex meta-features, can be found in the Appx.H.

Question 1: Is the model constructed by meta-predictor better than existing SOTA methods? Comprehensive forecasting results are presented in Table 2 and 3, where the best performances are highlighted in **red** and the second-best results are underlined. Compared with other state-of-the-art forecasters, TSGym demonstrates superior capability, especially in handling high-dimensional time series data. Its consistent top-ranking performance across multiple datasets underlines its robustness and effectiveness for complex multivariate forecasting tasks.

Table 2: Short-term forecasting task on M4. The results are averaged from several datasets under different sample intervals. See Table in Appendix for the full results.

Models	TSGym (ours)	TimeMixer	MICN	TimesNet	PatchTST	DLinear	Crossformer	Autoformer	SegRNN
OWA	**0.872**	0.884	0.984	0.907	0.965	0.922	8.856	1.273	1.007
SMAPE	12.013	**11.985**	13.025	12.199	12.848	12.511	>30	16.392	13.509
MASE	**1.575**	1.615	1.839	1.662	1.738	1.693	>10	2.317	1.823

Specifically, TSGym achieves the lowest MSE and MAE on a total of 11 occasions, reflecting its strong generalization ability over both medium and long forecasting horizons. While some baseline models, like DUET, PatchTST and SegRNN, occasionally show competitive results on certain datasets. As for short-term forecasting tasks, both TSGym and **TimeMixer** demonstrate competitive performance, with TSGym outperforming on most evaluation metrics.

Question 2: Is TSGym with lightweight architecture better than existing SOTA methods?

Table 3: Long-term forecasting task. The past sequence length is set as 36 for ILI and 96 for the others. All the results are averaged from 4 different prediction lengths, that is $\{24, 36, 48, 60\}$ for ILI and $\{96, 192, 336, 720\}$ for the others. See Table in Appendix for the full results.

Models	**TSGym (Ours)**		DUET [50]		TimeMixer [59]		MICN [58]		TimesNet [65]		PatchTST [46]		DLinear [72]		Crossformer [75]		Autoformer [35]		SegRNN [66]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.357**	**0.383**	0.407	0.409	0.384	0.399	0.402	0.429	0.432	0.430	0.390	0.404	0.404	0.407	0.501	0.501	0.532	0.496	0.388	0.404
ETTm2	**0.261**	**0.319**	0.296	0.338	0.277	0.325	0.342	0.391	0.296	0.334	0.288	0.334	0.349	0.399	1.487	0.789	0.330	0.368	0.273	0.322
ETTh1	0.426	0.440	0.433	0.437	0.448	0.438	0.589	0.537	0.474	0.464	0.454	0.449	0.465	0.461	0.544	0.520	0.492	0.485	**0.422**	**0.429**
ETTh2	**0.358**	**0.400**	0.380	0.403	0.383	0.406	0.585	0.530	0.415	0.424	0.385	0.409	0.566	0.520	1.552	0.908	0.446	0.460	0.374	0.405
ECL	**0.170**	0.265	0.179	**0.262**	0.185	0.273	0.186	0.297	0.219	0.314	0.209	0.298	0.225	0.319	0.193	0.289	0.234	0.340	0.216	0.302
Traffic	**0.435**	**0.313**	0.797	0.427	0.496	0.313	0.544	0.320	0.645	0.348	0.497	0.321	0.673	0.419	1.458	0.782	0.637	0.397	0.807	0.411
Weather	**0.229**	**0.268**	0.252	0.277	0.244	0.274	0.264	0.316	0.261	0.287	0.256	0.279	0.265	0.317	0.253	0.312	0.339	0.379	0.251	0.298
Exchange	0.410	0.431	**0.322**	**0.384**	0.359	0.402	0.346	0.422	0.405	0.437	0.381	0.412	0.346	0.414	0.904	0.695	0.506	0.500	0.408	0.423
ILI	2.233	1.015	2.640	1.018	4.502	1.557	2.938	1.178	**2.140**	0.907	2.160	**0.901**	4.367	1.540	4.311	1.396	3.156	1.207	4.305	1.397
1st Count	**11**		3		0		0		1		1		0		0		0		2	

In the previous section, we compared TSGym using the full component pool with SOTA and found that TSGym outperforms SOTA on 66.7% of the datasets evaluated. In this ablation experiment Table 4, we specifically compare the `-Transformer` configuration of TSGym with DUET. Remarkably, even after removing Transformer-related components from the TSGym component pool and retaining only the more computationally efficient MLP- and RNN-based models, TSGym still outperforms DUET on the majority of datasets. This demonstrates the robustness and efficiency of TSGym's architecture and highlights the strong predictive power of the simplified MLP-based design.

Question 3: Does the training strategies bring significant improvement for TSGym?

Following Table 4, we find that the introduction of the resampling method enhances TSGym's meta-predictor performance across 2 datasets, improving both robustness and accuracy. The `+AllPL` configuration, which trains on datasets with varying prediction lengths and transfers this knowledge to a test set with a single prediction length, further improves generalization, with the best performance observed on the ETTh1 dataset. Additionally, removing the Transformer component (`-Transformer`) leads to performance gains on certain datasets, suggesting that a simplified MLP- or RNN-based architecture can be more effective in specific scenarios. These results highlight the flexibility of TSGym's design and the potential benefits of customizing the component pool to suit dataset characteristics.

Table 4: Ablation study evaluates the removal of Transformer-based components and different training strategies, with results averaged over all prediction lengths, and the final row shows how often TSGym variants outperform DUET.

Models	**TSGym**		**-Transformer**		**+Resample**		**+AllPL**		**DUET**	
Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	**0.357**	**0.383**	0.363	0.388	0.361	0.384	0.367	0.391	0.407	0.409
ETTm2	0.261	**0.319**	0.274	0.330	**0.260**	0.320	0.265	0.320	0.296	0.338
ETTh1	0.426	0.440	0.441	0.450	0.432	0.440	**0.424**	**0.434**	0.433	0.437
ETTh2	0.358	0.400	0.358	0.401	**0.357**	**0.398**	0.361	0.404	0.380	0.403
ECL	**0.170**	0.265	0.174	0.273	0.170	0.265	0.185	0.277	0.179	**0.262**
Traffic	0.435	0.313	**0.415**	**0.293**	0.429	0.307	0.429	0.306	0.797	0.427
Weather	0.229	0.268	**0.226**	**0.265**	0.229	0.267	0.234	0.271	0.252	0.277
Exchange	0.410	0.431	0.409	0.435	0.399	0.430	0.377	0.411	**0.322**	**0.384**
ILI	2.233	**1.015**	2.437	1.053	2.698	1.114	**2.173**	1.020	2.640	1.018
Better Count	**14/18**		**12/18**		**12/18**		**12/18**			

Question 4: Does large time-series models bring significant improvement for TSGym?

Table 5 evaluates the impact of incorporating LLM and TSFM into the base TSGym framework. It is evident that both the introduction of LLM and TSFM consistently improve forecasting accuracy compared to the baseline TSGym configuration. This demonstrates the effectiveness of leveraging advanced model architectures and fusion strategies to further enhance the predictive performance of TSGym, particularly on complex multivariate time series datasets.

Table 5: Ablation study of TSGym incorporating LLM and TSFM in 4 datasets. The average results of all prediction lengths are listed here.

Models	**TSGym**		**+LLM**		**+TSFM**	
Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	0.442	0.440	0.442	0.438	**0.432**	**0.437**
ETTh2	0.411	0.429	**0.364**	**0.402**	0.376	0.409
Exchange	0.673	0.522	0.503	0.466	**0.374**	**0.411**
ILI	2.682	1.112	**2.470**	**1.063**	2.860	1.163

Question 5: Does smarter sampling strategy bring improvement for TSGym?

To further improve efficiency and reduce meta-training costs, we adopted Optuna, a smarter sampling strategy based on Bayesian optimization. Starting with 50 random trials as a cold

Table 6: **TSGym Performance Comparison Across Sampling Strategies and DUET. The average results of all prediction lengths are listed here.**

Models	TSGym (Optuna)		TSGym		DUET	
Metric	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.350**	**0.373**	0.357	0.383	0.407	0.409
ETTm2	**0.252**	**0.310**	0.261	0.319	0.296	0.338
ETTh1	0.429	0.440	**0.426**	0.440	0.433	**0.437**
ETTh2	**0.354**	**0.392**	0.358	0.400	0.380	0.403
ECL	**0.166**	**0.262**	0.170	0.265	0.179	**0.262**
Traffic	0.439	**0.295**	**0.435**	0.313	0.797	0.427
Weather	**0.229**	**0.258**	**0.229**	0.268	0.252	0.277
Exchange	0.457	0.446	0.410	0.431	**0.322**	**0.384**
ILI	3.194	1.234	**2.233**	**1.015**	2.640	1.018
1st Count	**11**		5		4	

start, Optuna then guided 50 additional trials using prior search knowledge, significantly enhancing search efficiency and identifying better model configurations with fewer evaluations. See Appx.J for details. Table 6 shows that the meta-learner trained on Optuna-selected samples achieves comparable or better performance than the one trained on random search. The only exception is the ILI dataset, where Optuna used a much smaller trial budget; performance is expected to improve with more trials.

5 Conclusions, Limitations, and Future Directions

To advance beyond holistic evaluations in multivariate time-series forecasting (MTSF), this paper introduced TSGym, a novel framework centered on fine-grained component analysis and the automated construction of specialized forecasting models. By systematically decomposing MTSF pipelines into design dimensions and choices informed by recent studies, TSGym uncovers crucial insights into component-level forecasting performance and leverages meta-learning method for the automated construction of customized models. Extensive experimental results indicate that the MTSF models constructed by the proposed TSGym significantly outperform current MTSF SOTA solutions—demonstrating the advantage of adaptively customizing models according to distinct data characteristics. Our results show that TSGym is highly effective, even without exhaustively covering all SOTA components, and TSGym is made publicly available to benefit the MTSF community.

Future efforts will focus on expanding TSGym's range of forecasting techniques with emerging techniques and refining its meta-learning capabilities by incorporating multi-objective optimization to balance predictive performance against computational costs, especially for large time-series models, while also broadening its applicability across diverse time series analysis tasks.

References

[1] Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[2] M. Abdallah, R. Rossi, K. Mahadik, S. Kim, H. Zhao, and S. Bagchi. Autoforecast: Automatic time-series forecasting model selection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 5–14, 2022.

[3] M. Abdallah, R. A. Rossi, K. Mahadik, S. Kim, H. Zhao, and S. Bagchi. Evaluation-free time-series forecasting model selection via meta-learning. *ACM Transactions on Knowledge Discovery from Data*, 2025.

[4] B. Abraham and J. Ledolter. *Statistical methods for forecasting*. John Wiley & Sons, 2009.

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[6] F. M. Alvarez, A. Troncoso, J. C. Riquelme, and J. S. A. Ruiz. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243, 2010.

[7] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[8] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[9] C. Bui, N. Pham, A. Vo, A. Tran, A. Nguyen, and T. Le. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In *6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6) 6*, pages 809–818. Springer, 2018.

[10] C. Catlin. Autots: Automated time series forecasting for python. `https://github.com/winedarksea/AutoTS`, 2020.

[11] C. Chang, W.-C. Peng, and T.-F. Chen. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *CoRR*, 2023.

[12] P. Chen, Y. ZHANG, Y. Cheng, Y. Shu, Y. Wang, Q. Wen, B. Yang, and C. Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[13] S.-A. Chen, C.-L. Li, S. O. Arik, N. C. Yoder, and T. Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

[14] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan. Towards spatio-temporal aware traffic time series forecasting. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2900–2913. IEEE, 2022.

[15] A. Das, W. Kong, A. Leach, S. K. Mathur, R. Sen, and R. Yu. Long-term forecasting with tiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

[16] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.

[17] W. Fan, P. Wang, D. Wang, D. Wang, Y. Zhou, and Y. Fu. Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7522–7529, 2023.

[18] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. 2020.

[19] R. Fischer and A. Saadallah. Autoxpcr: Automated multi-objective model selection for time series forecasting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 806–815, 2024.

[20] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski. MOMENT: A family of open time-series foundation models. In *Forty-first International Conference on Machine Learning*, 2024.

[21] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[22] H2O.ai. $package_or_module_title, monthyear.version_information.$

[23] A. D. Hartanto, Y. N. Kholik, and Y. Pristyanto. Stock price time series data forecasting using the light gradient boosting machine (lightgbm) model. *JOIV: International Journal on Informatics Visualization*, 7(4):2270–2279, 2023.

[24] M. Jiang, C. Hou, A. Zheng, S. Han, H. Huang, Q. Wen, X. Hu, and Y. Zhao. Adgym: Design choices for deep anomaly detection. *Advances in Neural Information Processing Systems*, 36:70179–70207, 2023.

[25] H. Jin, F. Chollet, Q. Song, and X. Hu. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6, 2023.

[26] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[27] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] M. Jin, Q. Wen, Y. Liang, C. Zhang, S. Xue, X. Wang, J. Zhang, Y. Wang, H. Chen, X. Li, S. Pan, V. S. Tseng, Y. Zheng, L. Chen, and H. Xiong. Large models for time series and spatio-temporal data: A survey and outlook. *CoRR*, abs/2310.10196, 2023.

[29] M. Jin, Y. Zhang, W. Chen, K. Zhang, Y. Liang, B. Yang, J. Wang, S. Pan, and Q. Wen. Position paper: What can large language models tell us about time series analysis. *arXiv e-prints*, pages arXiv–2402, 2024.

[30] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt. Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data*, 3:4, 2020.

[31] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021.

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[34] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[35] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, and H. Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. *arXiv preprint arXiv:2308.11200*, 2023.

[36] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, and Q. Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

[37] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. 2022.

[38] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[39] Y. Liu, C. Li, J. Wang, and M. Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. *Advances in neural information processing systems*, 36:12271–12290, 2023.

[40] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35:9881–9893, 2022.

[41] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Rethinking the stationarity in time series forecasting. In *NeurIPS*, 2022.

[42] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long. Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning*.

[43] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long. Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning*, 2024.

[44] R. P. Masini, M. C. Medeiros, and E. F. Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

[45] Microsoft. Neural Network Intelligence, 1 2021.

[46] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[47] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing, 2016.

[48] X. Qiu, H. Cheng, X. Wu, J. Hu, C. Guo, and B. Yang. A comprehensive survey of deep learning for multivariate time series forecasting: A channel strategy perspective. *arXiv preprint arXiv:2502.10721*, 2025.

[49] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng, et al. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *arXiv preprint arXiv:2403.20150*, 2024.

[50] X. Qiu, X. Wu, Y. Lin, C. Guo, J. Hu, and B. Yang. Duet: Dual clustering enhanced multivariate time series forecasting. KDD '25, page 1185–1196, New York, NY, USA, 2025. Association for Computing Machinery.

[51] M. Ruchte, A. Zela, J. Siems, J. Grabocka, and F. Hutter. Naslib: A modular and flexible neural architecture search library. `https://github.com/automl/NASLib`, 2020.

[52] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[53] Z. Shao, F. Wang, Y. Xu, W. Wei, C. Yu, Z. Zhang, D. Yao, T. Sun, G. Jin, X. Cao, et al. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[54] O. Shchur, A. C. Turkmen, N. Erickson, H. Shen, A. Shirkov, T. Hu, and B. Wang. Autogluon–timeseries: Automl for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*, pages 9–1. PMLR, 2023.

[55] O. Shchur, C. Turkmen, N. Erickson, H. Shen, A. Shirkov, T. Hu, and Y. Wang. AutoGluon-TimeSeries: AutoML for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*, 2023.

[56] M. Tan, M. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. Are language models actually useful for time series forecasting? *Advances in Neural Information Processing Systems*, 37:60162–60191, 2024.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[58] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*, 2023.

[59] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[60] Y. Wang, H. Wu, J. Dong, Y. Liu, M. Long, and J. Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024.

[61] Y. Wang, H. Wu, J. Dong, G. Qin, H. Zhang, Y. Liu, Y. Qiu, J. Wang, and M. Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[62] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4653–4660, 8 2021.

[63] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[64] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. C. H. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

[65] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

[66] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *NeurIPS*, 2021.

[67] X. Wu, X. Wu, B. Yang, L. Zhou, C. Guo, X. Qiu, J. Hu, Z. Sheng, and C. S. Jensen. Autocts++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting. *The VLDB Journal*, 33(5):1743–1770, 2024.

[68] P. T. Yamak, L. Yujian, and P. K. Gadosey. A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd international conference on algorithms, computing and artificial intelligence*, pages 49–55, 2019.

[69] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, and Z. Niu. Frequency-domain mlps are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36:76656–76679, 2023.

[70] Y. Yin and P. Shang. Forecasting traffic time series with multivariate predicting method. *Applied Mathematics and Computation*, 291:266–278, 2016.

[71] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? 2023.

[72] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[73] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[74] T. Zhang, Y. Zhang, W. Cao, J. Bian, X. Yi, S. Zheng, and J. Li. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures, 2022.

[75] Y. Zhang and J. Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

[76] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[77] T. Zhou, Z. Ma, Q. Wen, L. Sun, T. Yao, W. Yin, R. Jin, et al. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in neural information processing systems*, 35:12677–12690, 2022.

[78] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

[79] T. Zhou, P. Niu, L. Sun, R. Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We systematically present the research topic, background, methodology, and conclusions of this paper in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly state the limitations of this paper in the "Conclusions, Limitations, and Future Directions" section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: the paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have open-sourced all the code and data. Anyone can directly reproduce our experimental results using the predefined scripts provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have open-sourced all the code and data, including component-level evaluations and the meta-learning based automated selection model. All code and datasets are publicly available at `https://github.com/SUFE-AILAB/TSGym`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed descriptions of the datasets and experimental settings, the evaluation metrics used, the baselines, and the training hyperparameters of our proposed meta predictor in the Experiment Settings and Appendix sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We present the error bars in the box plots shown in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: provide sufficient information on the computer resources needed to reproduce the experiments in the Appendix Section.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics, adhering to ethical standards in methodology, transparency, and data usage.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In the Introduction section, we discuss the broad applicability and value of the research topic addressed in this paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper outlines the safeguards implemented for the responsible release of data and models in the experiment settings section , ensuring that they are protected against potential misuse.

    Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators and original owners of all assets (e.g., code, data, models) used in the paper are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide detailed descriptions of the code and experimental settings in the text.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

For further details, we provide more information in the Appendix, including the evaluated 10 datasets (§A), key modules (§B), compared baselines (§C), metrics mathematical formula (§D), system configuration (§E), ADGym comparison analysis (§F), the details of proposed TSGym (§G), and additional experimental results (§H).

A Dataset List

We conduct extensive evaluations on nine standard long-term forecasting benchmarks - four ETT variants (ETTh1, ETTh2, ETTm1, ETTm2), Electricity (abbreviated as ECL), Traffic, Weather, Exchange, and ILI, complemented by the M4 dataset for short-term forecasting tasks, with complete dataset specifications provided in Table A1.

Table A1: Data description of the 10 datasets included in TSGym.

Task	Dataset	Domain	Frequency	Lengths	Dim	Description
LTF	ETTh1	Electricity	1 hour	14,400	7	Power transformer 1, comprising seven indicators such as oil temperature and useful load
	ETTh2	Electricity	1 hour	14,400	7	Power transformer 2, comprising seven indicators such as oil temperature and useful load
	ETTm1	Electricity	15 mins	57,600	7	Power transformer 1, comprising seven indicators such as oil temperature and useful load
	ETTm2	Electricity	15 mins	57,600	7	Power transformer 2, comprising seven indicators such as oil temperature and useful load
	ECL	Electricity	1 hour	26,304	321	Electricity records the electricity consumption in kWh every 1 hour from 2012 to 2014
	Traffic	Traffic	1 hour	17,544	862	Road occupancy rates measured by 862 sensors on San Francisco Bay area freeways
	Weather	Environment	10 mins	52,696	21	Recorded every for the whole year 2020, which contains 21 meteorological indicators
	Exchange	Economic	1 day	7,588	8	ExchangeRate collects the daily exchange rates of eight countries
	ILI	Health	1 week	966	7	Recorded indicators of patients data from Centers for Disease Control and Prevention
STF	M4	Demographic, Finance, Industry, Macro, Micro and Other	Yearly Quarterly Monthly Weakly Daily Hourly	19-9933	100000	M4 competition dataset containing 100,000 unaligned time series with varying lengths and time periods

B Key Modules

Modern deep learning for MTSF utilizes several specialized modules to tackle non-stationarity, multi-scale dependencies, and inter-variable interactions. In this section, we analyze the design and efficacy of prevalent specialized modules adopted in state-of-the-art models (Fig. 1).

Normalization modules address temporal distribution shifts through adaptive statistical alignment. While z-score normalization employs fixed moments, modern techniques enhance adaptability: RevIN [31] introduces learnable affine transforms with reversible normalization/denormalization; Dish-TS [17] decouples inter-/intra-series distribution coefficients; Non-Stationary Transformer [40] integrates statistical moments into attention via de-stationary mechanisms. These methods balance stationarized modeling with inherent non-stationary dynamics.

Decomposition methods, standard in time series analysis, break down series into components like trend and seasonality to improve predictability and handle distribution shifts. **(1) Time-domain decomposition** utilizes moving average operations to isolate slowly-varying trends from high-frequency fluctuations that represent seasonality (e.g., DLinear [72], Autoformer, FEDformer). **(2) Frequency-domain decomposition** partitions series via Discrete Fourier Transform (DFT), assigning low-frequency spectra to trends and high-frequency bands to seasonality, which is applied in the Koopa [39] model.

Multi-Scale modeling addresses the inherent temporal hierarchy in time series data, where patterns manifest differently across various granularities (e.g., minute-level fluctuations vs. daily trends). Pyraformer [37] integrates multi-convolution kernels via pyramidal attention to establish hierarchical temporal dependencies. FEDformer [78] employs mixed experts to combine trend components from multiple pooling kernels with varying receptive fields, where larger kernels capture macro patterns while smaller ones preserve local details. TimeMixer [59] extends this paradigm through bidirectional mixing operations - upward propagation refines fine-scale seasonal features while downward aggregation consolidates coarse-scale trends. FiLM [77] dynamically adjusts temporal resolutions through learnable lookback windows, enabling adaptive focus on relevant historical contexts across scales. Crossformer [75] implements flexible patchsize configurations, where multi-granular patches independently model short-term fluctuations and long-term cycles through dimension-aware processing.

Temporal Tokenization strategies, originating from Transformers [60, 38] and now extended to RNNs [35], vary by temporal representation granularity: **(1) Point-wise** methods (e.g., Informer [76], Pyraformer [37]) process individual timestamps as tokens. They offer temporal precision but face quadratic complexity, requiring attention sparsification that may hinder long-range dependency capture. **(2) Patch-wise** strategies (e.g., PatchTST

[46]) aggregate local temporal segments into patches. Pathformer [12] similarly employs patch-based processing via adaptive multi-scale pathways. **(3) Series-wise** approaches (e.g., iTransformer [38]) construct global variate representations, enabling cross-variate modeling but risking temporal misalignment. TimeXer [61] uses hybrid tokenization: patch-level for endogenous variables and series-level for exogenous, bridged by a learnable global token.

Temporal Dependency Modeling captures dynamic inter-step dependencies through diverse architectural mechanisms, balancing local interactions and global patterns. Recurrent state transitions (e.g., LSTM) model sequential memory via gated memory cells; temporal convolutions (e.g., TCN [7]) construct multi-scale receptive fields using dilated kernels; attention mechanisms (e.g., Transformers) enable direct pairwise interactions across arbitrary time steps. Efficiency-driven innovations include sparse attention (Informer [76]), periodicity-based aggregation (Autoformer [66]), and state-space hybrids (Mamba [21]), achieving tractable long-range dependency modeling while preserving temporal fidelity.

Variate Correlation, fundamental to modeling critical correlations in multivariate time series forecasting (MTSF), operates through two primary paradigms [50]: **(1) Channel-Independent (CI) Strategy**: Processes channels independently with shared parameters (e.g., PatchTST [46]), ensuring robustness and efficiency but ignoring multivariate dependencies, limiting use with strong inter-channel interactions [48].

(2) Channel-Dependent (CD) Strategy: Integrates channel information via methods like channel-wise self-attention (iTransformer [38]) or MLP-based mixing (TSMixer [13]). This allows explicit dependency modeling but risks overfitting and struggles with noise in high dimensions.

C Compared Baselines

We systematically compare state-of-the-art forecasting models using the 6 architectural modules introduced in Section B. Table C2 presents the configuration of each baseline in terms of these modules. The "Notes" column provides concise annotations of each model's key methodological features, allowing for quick identification of the technical differentiators among the baselines.

D Metrics Mathematical Formula

The metrics used in this paper can be calculated as follows[65]:

$$\text{MSE} = \frac{1}{H}\sum_{i=1}^{H}(\mathbf{X}_i - \widehat{\mathbf{X}}_i)^2, \qquad \text{MAE} = \frac{1}{H}\sum_{i=1}^{H}|\mathbf{X}_i - \widehat{\mathbf{X}}_i|,$$

$$\text{SMAPE} = \frac{200}{H}\sum_{i=1}^{H}\frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{|\mathbf{X}_i| + |\widehat{\mathbf{X}}_i|}, \qquad \text{MAPE} = \frac{100}{H}\sum_{i=1}^{H}\frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{|\mathbf{X}_i|},$$

$$\text{MASE} = \frac{1}{H}\sum_{i=1}^{H}\frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{\frac{1}{H-m}\sum_{j=m+1}^{H}|\mathbf{X}_j - \mathbf{X}_{j-m}|}, \qquad \text{OWA} = \frac{1}{2}\left[\frac{\text{SMAPE}}{\text{SMAPE}_{\text{Naïve2}}} + \frac{\text{MASE}}{\text{MASE}_{\text{Naïve2}}}\right],$$

where m is the periodicity of the data. $\mathbf{X}, \widehat{\mathbf{X}} \in \mathbb{R}^{H\times C}$ are the ground truth and prediction results of the future with H time points and C dimensions. $\mathbf{X}_i$ means the i-th future time point.

E System Configuration

We conducted all experiments in the same experimental environment, which includes four NVIDIA A100 GPUs with 80GB and eight 40GB of memory. We saved overall experimental time by running experiments in parallel.

F Compared with ADGym

Compared with ADGym [24], TSGym exhibits the following differences and advantages:

(1) Broader model structure design choices. ADGym includes only MLP, autoencoder (AE), ResNet, and Transformer architectures, while TSGym provides an in-depth decoupling of different attention mechanisms within Transformers and incorporates two pre-trained large models: LLMs and TSFM. **(2) More diverse data processing design choices**. ADGym focuses solely on data augmentation and two normalization methods, whereas TSGym encompasses series sampling, series normalization, series decomposition, as well as various series encoding options. **(3) More complex meta-features**. The meta-features in ADGym include statistical metrics for tabular datasets, while TSGym considers multiple sequence characteristics across different channels in multivariate time series, such as distribution drift, sequence autocorrelation, and more. **(4) More standardized**

Table C2: Component Configurations of 27 Baseline Models

Backbone	Method	Normalization	Decomposition	Multi-Scale	Tokenizations	Temporal Dependency	Variate Correlation	Notes
RNN	SegRNN[35]	SubLast			Patch-wise	GRU	CI	Reduces iterations via patch-wise processing and parallel multi-step forecasting.
	Mamba[21]	Stat			Point-wise	Selective State Space Model	CD	Efficient model selectively propagating information without attention or MLP blocks.
CNN	SCINet[36]	Stat		TRUE	Point-wise	Conv1d	CD	Recursively downsamples, convolves, and interacts with data to capture complex temporal dynamics.
	MICN[58]		MA	TRUE	Point-wise	Conv1d	CD	Combines local features and global correlations using multi-scale convolutions with linear complexity.
	TimesNet[65]	Stat		TRUE	Point-wise	Conv2d	CD	Transforms 1D time series into 2D tensors to capture multi-periodicity and temporal variations.
MLP	FiLM[77]	RevIN		TRUE	Point-wise	Legendre Projection Unit	CD	Preserves historical info and reduces noise with Legendre and Fourier projections.
	LightTS[74]				Patch-wise	MLP	CD	Lightweight MLP model for multivariate forecasting, using continuous and interval sampling for efficiency.
	DLinear[72]		MA		Point-wise	MLP	CI/CD	Decomposes series into trend and seasonal components, then applies linear layers for improved forecasting.
	Koopa[39]	Stat	DFT		Point-wise, Patch-wise	MLP	CD	Uses Koopman theory to model non-stationary dynamics, handling time-variant and time-invariant components.
	TSMixer[13]				Point-wise	MLP	CD	Simple MLP model efficiently captures both time and feature dependencies for forecasting.
	FreTS[69]				Point-wise	Frequency-domain MLP	CI/CD	Uses frequency-domain MLPs to capture global dependencies and focus on key frequency components.
	TiDE[15]	Stat			Point-wise	MLP	CI	Fast MLP-based model for long-term forecasting, handling covariates and non-linear dependencies.
	TimeMixer[59]	RevIN	MA	TRUE	Point-wise	MLP	CI/CD	Fully MLP-based model, disentangles and mixes multi-scale temporal patterns.
Transformer	Reformer[1]				Point-wise	LSHSelf-Attention	CD	Memory-efficient Transformer with locality-sensitive hashing for faster training on long sequences.
	Informer[76]				Point-wise	ProbSparse-Attention	CD	Efficient Transformer with ProbSparse-Attention and a generative decoder for faster long-sequence forecasting.
	TFT[34]	Stat			Point-wise	Self-Attention	CD	High-performance, interpretable multi-horizon forecasting model combining recurrent layers for local processing and attention layers for long-term dependencies.
	Autoformer[66]		MA		Point-wise	Auto-Correlation	CD	Uses Auto-Correlation and decomposition for accurate long-term predictions.
	PyraFormer[37]			TRUE	Point-wise	Pyramid-Attention	CD	Captures temporal dependencies at multiple resolutions with constant signal path length.
	NSTransformer[40]	Stat			Point-wise	De-stationary Attention	CD	Restores non-stationary information through destationary attention for improved forecasting.
	ETSformer[64]		DFT		Point-wise	Exponential-Smoothing-Attention	CD	Integrates exponential smoothing and frequency attention for accuracy, efficiency, and interpretability.
	FEDformer[78]		MA	TRUE	Point-wise	AutoCorrelation	CD	Combines seasonal-trend decomposition with frequency-enhanced Transformer for efficient forecasting.
	Crossformer[75]			TRUE	Patch-wise	TwoStage-Attention	CD	Captures both temporal and cross-variable dependencies with two-stage attention.
	PatchTST[46]	Stat			Patch-wise	FullAttention	CI	Segments time series into patches and uses channel-independent embeddings.
	iTransformer[38]	Stat			Series-wise	FullAttention	CD	Redefines token embedding to treat time points as series-wise tokens for better multivariate modeling.
	TimeXer[61]	Stat			Series-wise	FullAttention	CD	Enhances forecasting by incorporating exogenous variables via patch-wise and variate-wise attention.
	PAttn[56]	Stat			Patch-wise	FullAttention	CI	Similar to PatchTST, uses attention-based patching for efficient forecasting without large language models.
	DUET[50]	RevIN	MA		Point-wise	FullAttention	CI/CD	Enhances multivariate forecasting by using Mixture of Experts (MOE) for temporal clustering and a frequency-domain similarity mask matrix for channel clustering.

Table F3: Compared with ADGym, TSGym covers a broader and more in-depth design space, as well as a more structured and extensive automated selection experiment.

	ADGym	TSGym
Design Dimensions	13	16
Design Space Size	195,9552	796,2624
Model Architectures	MLP,AE,ResNet,FTTransformer	MLP,RNN, Transformers, LLM, TSFM
Max of Data Samples	3000	57,600
Meta Feature Dimensions	200	1404
Baseline Methods	7	27

automated selection experiments. Due to time constraints, ADGym limits the sample size to fewer than 3000 samples, whereas TSGym imposes no such restriction, providing a larger-scale experimental design that leads to more solid experimental conclusions.

In summary, compared with ADGym, TSGym makes **significant progress and development in both components benchmarking and automated selection.** More details can be seen in table F3.

G Details of TSGym

In this section, we introduce detailed descriptions of the design choices, extracted meta-features and the trained meta-predictors.

G.1 More Details of Design Choices in TSGym.

We selected competitive components from the key modules of existing state-of-the-art (SOTA) works as our design choices. In Appx. B, we introduced the underlying principles of these components according to different design dimensions. While most of the individual components have demonstrated their effectiveness through ablation studies in their respective original papers, the interactions and synergies among them when combined have never been systematically explored. Notably, when assembling complete pipelines from different design choices, we automatically exclude incompatible combinations, such as pairing MLP-based architectures with diverse series attention modules. **To further enhance transparency, we have also created a new, detailed diagram Fig G1 that illustrates the workflow step-by-step through the pipeline.**

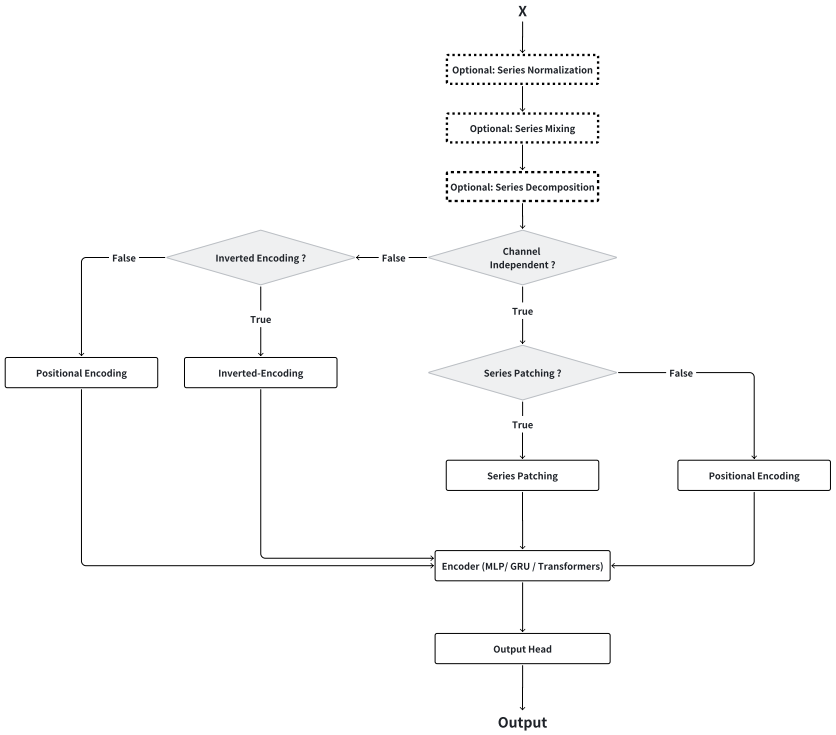


Figure G1: **TSGym pipeline framework. This diagram shows the component-based design of TSGym. The final TSGym structure is formed by combining different component options.**

G.2 Meta-features and Meta-predictors

Details and the selected list of meta-features. All meta-features in this paper integrate two complementary perspectives: (1) static characteristics extracted via TSFEL [49] spanning temporal, statistical, spectral, and fractal domains, and (2) dynamic behavioral metrics from TFB [8] to quantify temporal distribution shifts. In Section 4.2, we present the results of the meta-predictor trained on meta-features derived from static characteristics, which corresponds to the default setting in TSGym. Furthermore, in Fig. G2, we visualize the dimension-reduced meta-features across different datasets. In Table H10, we report the performance of the meta-predictor under various meta-feature configurations. The following categorizes these features with their analytical purposes (see Tables G4–G7 for implementation details):

- **Temporal features** (Table G4): Characterize sequential dynamics through trend detection, entropy analysis, and change-point statistics, preserving sensitivity to temporal ordering.
- **Statistical features** (Table G5): Capture distribution properties via central tendency (mean/median), dispersion (variance/IQR), and shape descriptors (skewness/kurtosis), invariant to observation order.

- **Spectral features** (Table G6): Decompose signals into frequency components using Fourier/wavelet transforms, identifying dominant periodicities and hidden oscillations.
- **Fractal features** (Table G7): Quantify multiscale complexity through fractal dimensions and Hurst exponents, reflecting self-similarity patterns across temporal resolutions.
- **Shifting Metric**: To complement static features, this TFB-derived metric measures temporal distribution drift via KL-divergence between adjacent windows. Values approaching 1 indicate severe shifts caused by external perturbations or systemic transitions, providing a diagnostic tool for non-stationary dynamics.

Table G4: **Temporal** Meta-feature Specifications

Feature	Description	Functionality
Absolute Energy	Computes the absolute energy of the signal.	Measures the total energy of the signal, often used to understand signal power and activity levels.
Area Under the Curve	Computes the area under the curve of the signal computed with the trapezoid rule.	Provides a measure of the overall signal amplitude or ""energy"" over time.
Autocorrelation	Calculates the first 1/e crossing of the autocorrelation function (ACF).	Measures the correlation of the signal with its own past values, useful for identifying repeating patterns.
Average Power	Computes the average power of the signal.	Averages the squared values of the signal, capturing its power over time.
Centroid	Computes the centroid along the time axis.	Indicates the ""center"" or ""balance point"" of the signal in time, providing insight into its distribution.
Signal Distance	Computes signal traveled distance.	Measures the total path length covered by the signal over time, capturing the extent of signal fluctuations.
Negative Turning	Computes number of negative turning points of the signal.	Counts the number of times the signal changes direction from positive to negative.
Neighbourhood Peaks	Computes the number of peaks from a defined neighbourhood of the signal.	Identifies the number of peak points within a specified window, useful for pattern detection.
Peak-to-Peak Distance	Computes the peak to peak distance.	Measures the time interval between successive peaks, indicating the period of oscillations.
Positive Turning	Computes number of positive turning points of the signal.	Counts the number of times the signal changes direction from negative to positive.
Root Mean Square	Computes root mean square of the signal.	Calculates the square root of the average squared values of the signal, often used as a measure of signal strength.
Slope	Computes the slope of the signal.	Measures the rate of change in the signal's amplitude over time, indicating trends or shifts.
Sum of Absolute Differences	Computes sum of absolute differences of the signal.	Measures the total variation in the signal by summing the absolute differences between consecutive values.
Zero-Crossing Rate	Computes Zero-crossing rate of the signal.	Counts how many times the signal crosses the zero axis, indicating its frequency and periodicity.

Details of the trained meta-predictors. For each design choice, we first use the LabelEncoder class from scikit-learn to convert it into a numerical class index. This index is then fed into an $nn.Embedding$ layer within our model to obtain a dense vector representation. These learned embeddings, along with other meta-features, subsequently form the input to the meta-predictor. The meta-predictor is optimized using Pearson loss to learn the relative performance ranks of different design choices, thereby emphasizing the linear correlation between predicted and actual rankings.

Moreover, we experimented with different training strategies to guide the meta-predictor in selecting the top-1 design pipelines.

(1) **+Resample**: Constraining the number of combinations from different datasets to be equal when training the meta-predictor.

(2) **+AllPL**: Training on datasets with varying prediction lengths and transfers this knowledge to a test set with a single prediction length.

(3) We train the meta-predictor using diverse meta features, including those generated by segmenting the datasets based on timestamps (**Sub**), those combining information from different time periods (**Whole**), and those designed to capture distributional shifts (**Delta**). The symbol **"+"** denotes the concatenation of multiple meta features.

We report the results of **+Resample** and **+AllPL** in Table 4, and the results of diverse meta-features in Table H10.

Table G5: **Statistical** Meta-feature Specifications

Feature	Description	Functionality
Maximum Value	Computes the maximum value of the signal.	Identifies the highest amplitude or peak value in the signal, useful for determining extreme values.
Mean Value	Computes mean value of the signal.	Calculates the average value of the signal, providing insight into its central tendency.
Median	Computes the median of the signal.	Finds the middle value of the signal when sorted, offering robustness to outliers.
Minimum Value	Computes the minimum value of the signal.	Identifies the lowest amplitude or trough value in the signal, useful for detecting minima.
Standard Deviation	Computes standard deviation (std) of the signal.	Measures the variation or spread of the signal values, indicating how much the signal deviates from the mean.
Variance	Computes variance of the signal.	Quantifies the spread of signal values, related to the square of the standard deviation.
Empirical Cumulative Distribution Function	Computes the values of ECDF along the time axis.	Provides a cumulative distribution function, representing the probability distribution of the signal values.
ECDF Percentile	Computes the percentile value of the ECDF.	Extracts specific percentiles from the cumulative distribution, useful for understanding the signal's quantiles.
ECDF Percentile Count	Computes the cumulative sum of samples that are less than the percentile.	Measures the number of samples falling below a given percentile, providing distribution insights.
ECDF Slope	Computes the slope of the ECDF between two percentiles.	Measures the steepness or rate of change in the cumulative distribution, indicating distribution sharpness.
Histogram Mode	Compute the mode of a histogram using a given number of bins.	Finds the most frequent value in the signal's histogram, representing the peak of the signal's distribution.
Interquartile Range	Computes interquartile range of the signal.	Measures the range between the 25th and 75th percentiles, indicating the spread of the central 50% of the signal values.
Kurtosis	Computes kurtosis of the signal.	Measures the ""tailedness"" of the signal distribution, indicating the presence of outliers or extreme values.
Mean Absolute Deviation	Computes mean absolute deviation of the signal.	Measures the average deviation of the signal values from the mean, providing an indication of signal variability.
Mean Absolute Difference	Computes mean absolute differences of the signal.	Calculates the average of absolute differences between successive signal values, reflecting the signal's smoothness.
Mean Difference	Computes mean of differences of the signal.	Computes the average of the first-order differences, used to measure overall signal change.
Median Absolute Deviation	Computes median absolute deviation of the signal.	Measures the spread of the signal values around the median, offering a robust measure of variability.
Median Absolute Difference	Computes median absolute differences of the signal.	Similar to mean absolute difference but based on the median, used to assess signal smoothness.
Median Difference	Computes median of differences of the signal.	Calculates the median of first-order differences, providing insights into signal trend stability.
Skewness	Computes skewness of the signal.	Measures the asymmetry of the signal's distribution, indicating whether it is skewed towards higher or lower values.

Table G6: **Spectral** Meta-feature Specifications

Feature	Description	Functionality
Entropy	Computes the entropy of the signal using Shannon Entropy.	Quantifies the uncertainty or randomness in the signal, offering insights into its complexity.
Fundamental Frequency	Computes the fundamental frequency of the signal.	Identifies the primary frequency at which the signal oscillates, crucial for detecting periodic behaviors.
Human Range Energy	Computes the human range energy ratio.	Measures the energy in the human audible range, useful for identifying signals relevant to human hearing.
Linear Prediction Cepstral Coefficients	Computes the linear prediction cepstral coefficients.	Extracts features related to the signal's frequency components, commonly used in speech and audio processing.
Maximum Frequency	Computes maximum frequency of the signal.	Identifies the highest frequency component of the signal, providing insight into its frequency range.
Maximum Power Spectrum	Computes maximum power spectrum density of the signal.	Measures the peak value in the power spectral density, identifying dominant frequencies in the signal.
Median Frequency	Computes median frequency of the signal.	Identifies the frequency that divides the signal's power spectrum into two equal halves.
Mel-Frequency Cepstral Coefficients	Computes the MEL cepstral coefficients.	Used to extract features representing the spectral characteristics of the signal, primarily used in speech analysis.
Multiscale Entropy	Computes the Multiscale entropy (MSE) of the signal, that performs entropy analysis over multiple scales.	Quantifies the signal's complexity at different scales, useful for detecting nonlinear temporal behaviors.
Power Bandwidth	Computes power spectrum density bandwidth of the signal.	Measures the width of the frequency band where the majority of the signal's power is concentrated.
Spectral Centroid	Barycenter of the spectrum.	Identifies the ""center"" of the signal's frequency spectrum, used in sound and audio analysis.
Spectral Decrease	Represents the amount of decreasing of the spectra amplitude.	Measures how rapidly the spectral amplitude decreases across frequency, useful for identifying spectral roll-off.
Spectral Distance	Computes the signal spectral distance.	Quantifies the difference between the signal's spectrum and a reference, helpful in pattern recognition.
Spectral Entropy	Computes the spectral entropy of the signal based on Fourier transform.	Measures the randomness or complexity in the frequency domain of the signal.
Spectral Kurtosis	Measures the flatness of a distribution around its mean value.	Quantifies the tail heaviness of the signal's frequency distribution, identifying outliers or abnormal distributions.
Spectral Positive Turning	Computes number of positive turning points of the fft magnitude signal.	Counts the points where the signal's Fourier transform changes direction from negative to positive.
Spectral Roll-Off	Computes the spectral roll-off of the signal.	Measures the frequency below which a specified percentage of the total spectral energy is contained.
Spectral Roll-On	Computes the spectral roll-on of the signal.	Similar to roll-off but identifies the frequency above which a specified amount of energy is concentrated.
Spectral Skewness	Measures the asymmetry of a distribution around its mean value.	Measures the skew in the signal's frequency distribution, highlighting the presence of spectral biases.
Spectral Slope	Computes the spectral slope.	Quantifies the slope of the power spectral density, often used to distinguish between harmonic and non-harmonic signals.
Spectral Spread	Measures the spread of the spectrum around its mean value.	Measures the dispersion or spread of the signal's spectral energy.
Spectral Variation	Computes the amount of variation of the spectrum along time.	Quantifies how much the frequency content of the signal changes over time.
Spectrogram Mean Coefficients	Calculates the average power spectral density (PSD) for each frequency throughout the entire signal.	Averages the power spectral density across all time intervals, capturing the signal's overall spectral energy distribution.
Wavelet Absolute Mean	Computes CWT absolute mean value of each wavelet scale.	Measures the average wavelet transform magnitude across scales, useful for detecting changes in signal frequency.
Wavelet Energy	Computes CWT energy of each wavelet scale.	Quantifies the energy at each wavelet scale, reflecting the signal's energy distribution across frequencies.
Wavelet Entropy	Computes CWT entropy of the signal.	Measures the complexity or unpredictability of the signal at different wavelet scales.
Wavelet Standard Deviation	Computes CWT std value of each wavelet scale.	Measures the variation or spread of the wavelet transform across different scales.
Wavelet Variance	Computes CWT variance value of each wavelet scale.	Quantifies the dispersion of the signal at different wavelet scales.

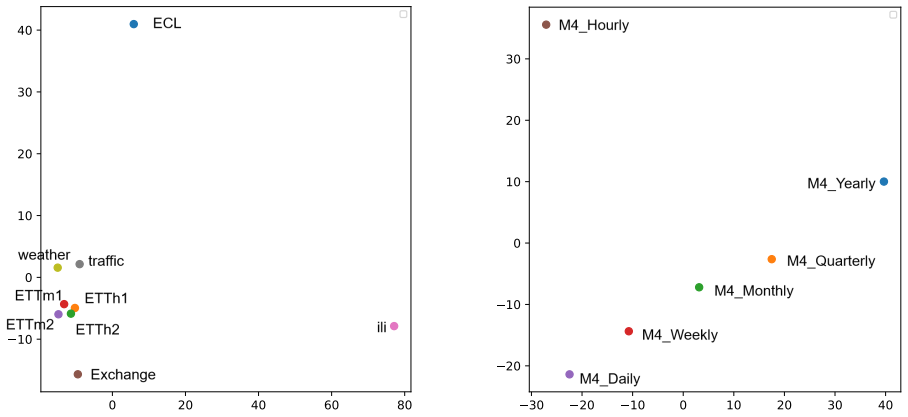
Table G7: **Fractal** Meta-feature Specifications

Feature	Description	Functionality
Detrended Fluctuation Analysis	Computes the Detrended Fluctuation Analysis (DFA) of the signal.	Measures long-range correlations and self-similarity in the signal, used for identifying fractal behavior.
Higuchi Fractal Dimension	Computes the fractal dimension of a signal using Higuchi's method (HFD).	Measures the complexity of the signal's pattern by calculating its fractal dimension.
Hurst Exponent	Computes the Hurst exponent of the signal through the Rescaled range (R/S) analysis.	Measures the long-term memory or persistence in the signal, useful for identifying trends and randomness.
Lempel-Ziv Complexity	Computes the Lempel-Ziv's (LZ) complexity index, normalized by the signal's length.	Quantifies the randomness or predictability of the signal based on its compressibility.
Maximum Fractal Length	Computes the Maximum Fractal Length (MFL) of the signal.	Measures the fractal dimension at the smallest scale of the signal, reflecting its intricate pattern complexity.
Petrosian Fractal Dimension	Computes the Petrosian Fractal Dimension of a signal.	Measures the signal's fractal dimension based on its variation across different scales.

Table H8: Full results for the long-term forecasting task. All the results are averaged from 4 different prediction lengths, that is $\{24, 36, 48, 60\}$ for ILI and $\{96, 192, 336, 720\}$ for the others.

Models	**TSGym (Ours)**		DUET [50]		TimeMixer [59]		MICN [58]		TimesNet [65]		PatchTST [46]		DLinear [72]		Crossformer [75]		Autoformer [66]		SegRNN [35]		Mamba [21]		iTransformer [38]		TimeXer [61]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.360**	**0.384**	0.407	0.409	0.384	0.399	0.402	0.429	0.432	0.430	0.390	0.404	0.404	0.407	0.501	0.501	0.532	0.496	0.388	0.404	0.501	0.466	0.414	0.415	0.386	0.400
ETTm2	0.265	**0.322**	0.296	0.338	0.277	0.325	0.342	0.391	0.296	0.334	0.288	0.334	0.349	0.399	1.487	0.789	0.330	0.368	0.273	0.322	0.356	0.370	0.290	0.332	0.279	0.325
ETTh1	0.425	0.434	0.433	0.437	0.448	0.438	0.589	0.537	0.474	0.464	0.454	0.449	0.465	0.461	0.544	0.520	0.492	0.485	**0.422**	**0.429**	0.544	0.504	0.462	0.452	0.446	0.443
ETTh2	**0.371**	0.406	0.380	0.403	0.383	0.406	0.585	0.530	0.415	0.424	0.385	0.409	0.566	0.520	1.552	0.908	0.446	0.460	0.374	0.405	0.465	0.448	0.382	0.406	0.372	**0.399**
ECL	**0.179**	0.275	0.179	**0.262**	0.185	0.273	0.186	0.297	0.219	0.314	0.209	0.298	0.225	0.319	0.193	0.289	0.234	0.340	0.216	0.302	0.209	0.312	0.190	0.277	0.191	0.286
Traffic	**0.434**	**0.310**	0.797	0.427	0.496	0.313	0.544	0.320	0.645	0.348	0.497	0.321	0.673	0.419	1.458	0.782	0.637	0.397	0.807	0.411	0.679	0.380	0.474	0.318	0.509	0.333
Weather	**0.229**	**0.267**	0.252	0.277	0.244	0.274	0.264	0.316	0.261	0.287	0.256	0.279	0.265	0.317	0.253	0.312	0.339	0.379	0.251	0.298	0.291	0.315	0.259	0.280	0.243	0.273
Exchange	0.392	0.418	**0.322**	**0.384**	0.359	0.402	0.346	0.422	0.405	0.437	0.381	0.412	0.346	0.414	0.904	0.695	0.506	0.500	0.408	0.423	0.714	0.562	0.369	0.410	0.410	0.424
ILI	2.345	1.053	2.640	1.018	4.502	1.557	2.938	1.178	2.140	0.907	2.160	**0.901**	4.367	1.540	4.311	1.396	3.156	1.207	4.305	1.397	3.729	1.335	2.305	0.974	2.633	1.034
1st Count	9		3		0		0		0		1		0		0		0		2		0		0		1	

Models	PAttn [56]		Koopa [39]		TSMixer [13]		FreTS [69]		Pyraformer [37]		Nonstationary [41]		ETSformer [64]		FEDformer [78]		SCINet [36]		LightTS [74]		Informer [76]		Transformer [57]		Reformer [1]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.384	0.399	0.367	0.396	0.527	0.512	0.409	0.417	0.695	0.593	0.509	0.467	0.636	0.592	0.438	0.450	0.409	0.412	0.438	0.445	0.969	0.736	0.836	0.678	0.998	0.723
ETTm2	0.291	0.336	**0.264**	0.327	1.030	0.750	0.336	0.378	1.565	0.876	0.412	0.398	1.381	0.807	0.301	0.348	0.294	0.335	0.432	0.448	1.504	0.878	1.454	0.851	1.856	0.996
ETTh1	0.468	0.454	0.472	0.471	0.615	0.579	0.476	0.464	0.814	0.692	0.610	0.543	0.750	0.651	0.448	0.461	0.520	0.488	0.530	0.505	1.057	0.798	0.930	0.768	0.973	0.739
ETTh2	0.386	0.412	0.388	0.423	2.160	1.220	0.548	0.514	3.776	1.557	0.552	0.505	0.572	0.534	0.427	0.446	0.428	0.440	0.633	0.551	4.535	1.745	2.976	1.369	2.487	1.238
ECL	0.205	0.286	0.219	0.319	0.229	0.337	0.209	0.296	0.295	0.387	0.194	0.296	0.275	0.370	0.225	0.336	0.220	0.323	0.243	0.344	0.369	0.444	0.273	0.367	0.324	0.404
Traffic	0.513	0.328	0.595	0.413	0.599	0.403	0.597	0.377	0.697	0.391	0.642	0.351	1.035	0.584	0.615	0.379	0.654	0.419	0.656	0.428	0.830	0.464	0.708	0.384	0.694	0.380
Weather	0.257	0.280	0.230	0.271	0.242	0.301	0.255	0.299	0.284	0.349	0.289	0.312	0.365	0.424	0.315	0.369	0.256	0.283	0.245	0.295	0.572	0.523	0.599	0.531	0.475	0.472
Exchange	0.365	0.407	0.610	0.516	0.487	0.546	0.442	0.453	1.183	0.855	0.557	0.490	0.361	0.416	0.520	0.502	0.374	0.418	0.486	0.493	1.548	0.997	1.379	0.921	1.612	1.044
ILI	2.359	0.975	**2.064**	0.912	5.617	1.680	3.447	1.279	4.691	1.442	2.592	1.012	4.046	1.419	3.088	1.214	6.505	1.853	7.078	1.975	5.035	1.539	4.682	1.448	4.211	1.350
1st Count	0		2		0		0		0		0		0		0		0		0		0		0		0	



(a) PCA projection of meta-features for 9 long-term forecasting datasets

(b) PCA projection of meta-features for 6 short-term forecasting datasets

Figure G2: Distributions of meta-features after PCA dimensionality reduction, comparing datasets for long-term and short-term time series forecasting tasks.

H Additional Experimental Results

H.1 Comprehensive Results of TSGym Against State-of-the-Art Methods

Due to space limitations in the main text, here we provide complete experimental comparisons for both long-term and short-term forecasting tasks. Table H.1 details the full long-term forecasting performance across all prediction horizons, while Table H.1 presents the comprehensive short-term forecasting results. Following standard benchmarking conventions, we highlight top-performing methods in **red** and second-best results with underlined formatting. These extensive evaluations consistently validate TSGym's competitive performance across diverse temporal prediction scenarios. In addition, we investigate the impact of different meta-feature configurations through controlled ablation studies. As demonstrated in Table H10, no individual meta-feature configuration exhibits consistent superiority across all datasets.

H.2 Additional Results of Large Evaluations on Design Choices

To systematically evaluate our architectural decisions, we conduct detailed ablation studies focusing on 17 component-level analyses, presented separately in Tables H11–H12 for clarity and due to space constraints. These comparative experiments assess the performance impact of different design choices for each component across nine datasets in the long-term forecasting task. **Bolded** values indicate the best-performing configuration for each dataset, while the summary row highlights the most frequently superior design choices, with **red-bolded**

Table H9: Full results for the short-term forecasting task in the M4 dataset. ∗. in the Transformers indicates the name of ∗former.

Models		TSGym (Ours)	TimeMixer [59]	MICN [58]	TimesNet [65]	PatchTST [46]	DLinear [72]	Cross. [75]	Auto. [66]	SegRNN [35]	Mamba [21]	iTrans. [38]	TimeXer [61]	PAttn [56]	TSMixer [13]	FreTS [69]	Pyra. [37]	ETS. [64]	FED. [78]	SCINet [36]	LightTS [74]	In. [76]	Trans. [57]	Re. [1]	TiDE [15]	FiLM [77]
Yearly	OWA	**0.779**	0.789	0.873	0.784	0.798	0.843	4.407	1.019	0.858	0.790	0.807	0.797	0.823	0.798	0.800	0.883	0.982	0.806	0.801	0.795	0.887	4.406	0.829	0.807	0.806
	sMAPE	**13.286**	13.467	14.586	13.344	13.606	14.402	71.464	17.294	14.323	13.388	13.681	13.551	13.893	13.539	13.579	14.967	16.105	13.631	13.573	13.516	15.086	69.405	14.064	14.006	13.988
	MASE	**2.960**	3.000	3.392	2.983	3.033	3.198	17.649	3.897	3.335	3.021	3.090	3.043	3.162	3.051	3.056	3.377	3.888	3.088	3.065	3.030	3.382	18.144	3.166	3.011	3.007
Quarterly	OWA	0.891	0.911	1.025	**0.886**	0.975	0.928	8.208	1.290	1.009	0.912	1.050	0.942	0.897	0.903	0.908	1.002	1.312	0.940	0.922	0.886	1.248	8.190	1.007	0.959	0.960
	sMAPE	10.232	10.313	11.427	**10.063**	10.975	10.500	74.297	14.085	11.193	10.320	11.752	10.536	10.203	10.218	10.329	11.259	13.600	10.655	10.426	10.166	13.644	73.944	11.324	10.719	10.742
	MASE	1.169	1.215	1.388	1.176	1.306	1.238	13.260	1.784	1.374	1.217	1.416	1.272	1.189	1.204	1.205	1.345	1.906	1.252	1.229	**1.163**	1.723	13.256	1.352	1.295	1.296
Monthly	OWA	**0.863**	0.895	0.986	0.939	1.020	0.936	7.637	1.369	1.074	0.931	0.982	0.944	0.951	0.898	0.915	1.043	1.281	1.001	0.924	0.884	1.151	7.668	1.448	0.942	0.942
	sMAPE	**12.570**	12.823	13.798	13.314	14.156	13.384	68.873	18.132	15.052	13.152	13.737	13.254	13.421	12.865	13.059	14.666	15.449	14.112	13.146	12.717	15.806	69.992	18.782	13.381	13.352
	MASE	**0.909**	0.959	1.080	1.015	1.126	1.004	11.165	1.576	1.175	1.010	1.076	1.030	1.034	0.962	0.984	1.138	1.586	1.090	0.996	0.943	1.283	11.149	1.694	1.017	1.019
Weekly	OWA	**0.983**	1.266	1.500	1.310	1.035	1.461	28.636	1.575	0.998	1.449	1.424	1.179	1.124	1.525	1.286	1.313	1.024	1.094	1.438	1.340	1.537	28.093	1.473	1.451	1.280
	sMAPE	9.467	11.555	11.790	11.569	9.546	11.805	198.371	12.727	**9.149**	12.495	12.050	10.455	10.326	12.364	11.394	11.742	9.363	9.635	12.757	12.060	11.967	191.424	11.522	12.425	11.539
	MASE	**2.593**	3.529	4.760	3.770	2.857	4.534	98.925	4.892	2.771	4.262	4.254	3.379	3.111	4.723	3.688	3.732	2.848	3.155	4.122	3.789	4.911	98.015	4.690	4.295	3.609
Daily	OWA	**0.982**	1.040	1.245	1.079	1.018	1.090	48.627	1.418	0.988	1.130	1.191	1.047	1.011	1.230	1.088	1.111	1.061	0.998	1.082	1.086	1.235	29.620	1.496	1.106	1.082
	sMAPE	**3.005**	3.162	3.786	3.294	3.103	3.319	179.226	4.248	3.009	3.417	3.607	3.198	3.089	3.677	3.304	3.398	3.216	3.060	3.288	3.313	3.727	99.709	4.521	3.342	3.267
	MASE	**3.205**	3.418	4.089	3.530	3.332	3.577	125.892	4.722	3.237	3.732	3.928	3.423	3.303	4.109	3.580	3.626	3.497	3.247	3.552	3.557	4.086	86.873	4.941	3.653	3.580
Hourly	OWA	0.902	1.372	1.526	3.171	2.625	1.040	11.691	1.623	1.704	**0.730**	1.214	1.636	1.683	1.243	1.393	2.315	1.201	1.126	1.372	1.183	3.166	6.498	3.204	1.231	1.445
	sMAPE	18.203	19.994	24.631	34.626	29.980	17.260	128.419	25.407	34.523	**14.944**	19.573	21.244	23.431	19.809	20.805	26.112	19.751	18.858	24.196	21.382	34.038	99.324	34.755	20.828	21.088
	MASE	1.947	3.966	4.100	10.680	8.667	2.732	39.269	4.465	3.663	**1.549**	3.266	5.067	5.009	3.372	3.964	7.685	3.178	2.937	3.420	2.881	10.730	18.188	10.821	3.183	4.172
Average	OWA	**0.856**	0.884	0.984	0.907	0.965	0.922	8.856	1.273	1.007	0.903	0.969	0.918	0.915	0.894	0.897	1.005	1.209	0.942	0.906	0.875	1.127	8.039	1.209	0.925	0.924
	sMAPE	**11.781**	11.985	13.025	12.199	12.848	12.511	76.147	16.392	13.509	12.120	12.838	12.268	12.351	12.023	12.137	13.478	14.635	12.708	12.219	11.925	14.673	72.619	15.344	12.489	12.471
	MASE	**1.551**	1.615	1.839	1.662	1.738	1.693	18.440	2.317	1.823	1.651	1.762	1.677	1.680	1.657	1.645	1.844	2.284	1.695	1.657	1.605	2.042	16.805	2.136	1.674	1.673
1st Count		**14**	0	0	2	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table H10: Effects of different meta feature settings on the long-term forecasting task. All metric values are averaged across different prediction lengths. For more details about meta features, refer to section G.2.

Models	Whole+Sub+Delta		Sub		Sub+Delta		Delta		Whole(default)		DUET	
Metric	mse	mae	mse	mae	mse	mae	mse	mae	mse	mae	mse	mae
ETTm1	0.357	0.383	0.363	0.388	**0.354**	**0.380**	0.377	0.405	0.357	0.383	0.407	0.409
ETTm2	0.273	0.329	0.269	0.329	0.266	0.324	0.380	0.389	**0.261**	**0.319**	0.296	0.338
ETTh1	**0.417**	**0.429**	0.433	0.435	0.418	0.433	0.558	0.496	0.426	0.440	0.433	0.437
ETTh2	0.375	0.407	0.362	0.402	0.360	**0.398**	1.256	0.744	**0.358**	0.400	0.380	0.403
ECL	0.172	0.266	0.171	0.269	0.176	0.270	0.182	0.278	**0.170**	0.265	0.179	**0.262**
Traffic	0.433	0.309	0.437	0.313	**0.432**	**0.308**	0.587	0.368	0.435	0.313	0.797	0.427
Weather	0.239	0.274	**0.228**	**0.266**	0.233	0.270	0.263	0.310	0.229	0.268	0.252	0.277
Exchange	0.406	0.429	0.408	0.429	0.404	0.428	0.761	0.622	0.410	0.431	**0.322**	**0.384**
ILI	2.401	1.030	3.099	1.195	2.855	1.141	2.814	1.125	**2.233**	**1.015**	2.640	1.018

Table H11: Long-term Forecasting Performance of Different Design Choices – Part I (6 Components). Performance of various configurations for 6 Components across multiple datasets, evaluated using best MSE, median, and IQR. **Bolded** entries indicate the best-performing hyperparameter for each dataset. The last row shows the number of times each configuration achieved the best result, with **red-bolded** values highlighting the most frequently superior design.

		x_mark		multi-granularity		Normalization				Decomposition				Channel-independent		Tokenization		
dataset	stat	False	True	False	True	DishTS	None	RevIN	Stat	DFT	MA	MoEMA	None	False	True	inverted-encoding	series-encoding	series-patching
ETTm1	Best	0.352	**0.35**	**0.349**	0.352	0.36	0.362	**0.351**	0.353	**0.351**	0.352	0.354	0.354	0.354	**0.35**	0.354	0.352	**0.351**
	Median	**0.423**	0.452	**0.428**	0.455	0.528	0.583	0.406	**0.405**	0.454	**0.406**	0.45	0.476	0.469	**0.389**	0.404	0.485	**0.384**
	IQR	**0.145**	0.179	**0.155**	0.186	0.212	0.235	**0.097**	0.097	0.159	**0.131**	0.169	0.179	0.195	**0.107**	0.132	0.207	**0.092**
ETTm2	Best	**0.255**	0.255	**0.254**	0.255	0.272	0.277	**0.253**	0.255	0.257	**0.256**	0.258	0.256	0.259	**0.253**	0.259	0.257	**0.254**
	Median	**0.367**	0.367	**0.356**	0.406	0.76	1.07	**0.297**	0.3	**0.353**	0.381	0.384	0.38	0.408	**0.307**	**0.305**	0.452	0.307
	IQR	**0.579**	0.782	**0.428**	0.947	0.746	1.267	0.045	**0.036**	0.552	0.769	0.681	**0.51**	0.871	**0.154**	0.226	0.963	**0.177**
ETTh1	Best	**0.401**	0.408	**0.403**	0.405	0.44	0.433	0.404	**0.402**	0.407	0.41	**0.405**	0.411	0.412	**0.401**	0.412	0.414	**0.401**
	Median	**0.491**	0.491	**0.484**	0.51	0.545	0.632	0.468	**0.462**	0.494	0.489	0.495	**0.486**	0.511	**0.462**	0.478	0.52	**0.456**
	IQR	**0.125**	0.126	**0.098**	0.183	0.207	0.371	**0.05**	0.05	0.141	**0.098**	0.157	0.118	0.202	**0.042**	0.062	0.241	**0.037**
ETTh2	Best	**0.326**	0.334	0.339	**0.326**	0.41	0.402	**0.325**	0.337	0.338	**0.327**	0.344	0.339	**0.327**	0.341	0.348	**0.325**	0.347
	Median	**0.444**	0.467	**0.449**	0.455	1.188	1.764	**0.392**	0.397	**0.43**	0.453	0.547	0.438	0.532	**0.394**	0.495	0.468	**0.388**
	IQR	0.939	**0.732**	**0.65**	1.74	1.657	2.855	**0.044**	0.048	**0.494**	1.035	1.105	0.726	1.513	**0.189**	0.418	1.924	**0.217**
ECL	Best	0.159	**0.157**	**0.157**	0.159	0.159	0.16	0.159	**0.157**	0.158	0.163	0.161	**0.157**	**0.157**	0.163	**0.157**	0.158	0.164
	Median	0.208	**0.204**	**0.204**	0.208	0.219	0.229	**0.191**	0.191	0.206	0.209	0.206	**0.202**	0.207	**0.202**	0.194	0.213	**0.19**
	IQR	**0.056**	0.057	0.058	**0.055**	0.053	0.059	**0.035**	0.052	0.065	**0.053**	0.054	0.056	0.057	**0.055**	**0.051**	0.061	0.051
traffic	Best	**0.394**	0.396	0.398	**0.394**	0.411	0.441	0.398	**0.394**	0.398	0.4	**0.394**	0.4	**0.394**	0.409	0.399	**0.394**	0.409
	Median	**0.555**	0.599	**0.576**	0.581	0.546	0.657	0.542	**0.496**	0.607	0.575	**0.56**	0.561	**0.568**	0.627	**0.531**	0.6	0.607
	IQR	**0.19**	0.199	0.208	**0.183**	0.164	**0.12**	0.207	0.195	**0.185**	0.19	0.203	0.198	0.196	**0.195**	0.19	0.191	**0.183**
weather	Best	0.223	**0.22**	**0.22**	0.222	0.225	0.225	**0.22**	0.224	0.226	0.223	**0.22**	0.221	**0.22**	0.22	**0.22**	0.22	0.223
	Median	**0.261**	0.272	**0.259**	0.274	0.267	0.301	**0.255**	0.256	0.265	0.27	**0.261**	0.263	0.273	**0.246**	0.247	0.281	**0.246**
	IQR	**0.041**	0.079	**0.047**	0.065	0.059	0.21	**0.033**	0.04	**0.047**	0.053	0.052	0.055	0.062	**0.035**	0.034	0.073	**0.033**
Exchange	Best	0.245	**0.237**	**0.239**	0.242	**0.24**	0.25	0.351	0.337	0.243	**0.239**	0.246	0.242	0.24	**0.238**	0.24	0.245	**0.238**
	Median	**0.493**	0.502	**0.462**	0.548	0.674	0.93	0.432	**0.415**	**0.472**	0.519	0.495	0.507	0.569	**0.394**	0.415	0.582	**0.395**
	IQR	**0.43**	0.471	**0.434**	0.491	0.869	0.855	**0.164**	0.168	**0.386**	0.546	0.43	0.466	0.595	**0.144**	0.253	0.613	**0.149**
ili	Best	1.596	**1.546**	**1.562**	1.576	1.763	2.351	1.584	**1.555**	1.673	1.599	1.581	**1.573**	**1.545**	1.745	1.583	**1.548**	1.745
	Median	**2.813**	2.883	2.881	**2.797**	2.785	4.416	**2.486**	2.493	2.878	**2.784**	2.859	2.883	**2.796**	3.043	2.865	2.844	**2.819**
	IQR	**1.621**	1.698	**1.656**	1.681	1.152	0.975	**0.742**	0.764	1.612	**1.596**	1.701	1.739	**1.665**	1.693	1.757	1.68	**1.442**
1st Count		**20**	7	**22**	5	1	1	**16**	9	8	**9**	5	5	8	**19**	5	3	**19**

entries denoting the dominant configurations. This fine-grained analysis offers empirical insights to guide component selection in time-series forecasting systems.

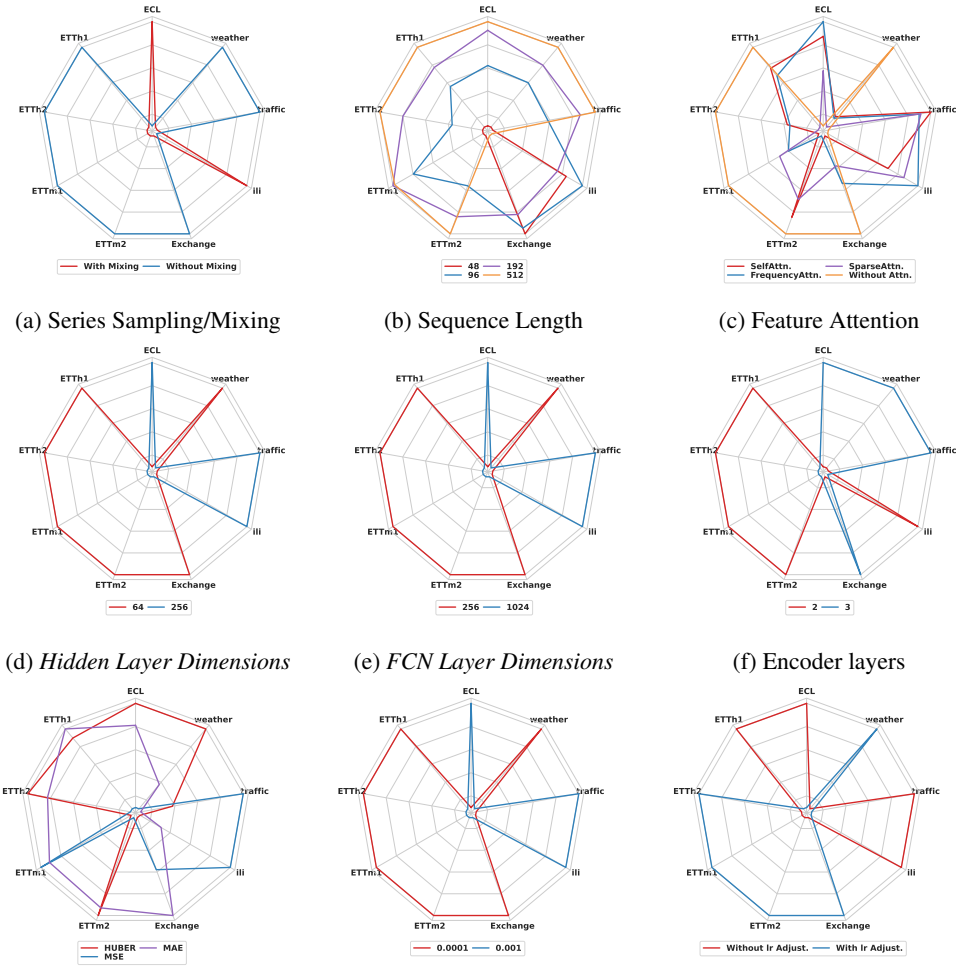
Table H12: Long-term Forecasting Performance of Different Design Choices – Part II (4 Components) and Part II (7 Components). Same structure and evaluation metrics as Table H11.

(a) Part II – 4 Components (Backbone, Attention, etc.)

		Backbone			Attention						Feature-Attention				Sequence Length			
dataset	stat	GRU	MLP	Trans-former	auto-cor-relation	de-stationary-attention	frequency-enhanced-attention	null	self-attention	sparse-attention	frequency-enhanced-attention	null	self-attention	sparse-attention	192	48	512	96
ETTm1	Best	0.352	0.352	**0.351**	0.359	0.382	0.354	**0.35**	0.359	0.354	0.356	**0.35**	0.355	0.36	0.352	0.476	**0.349**	0.38
	Median	0.457	**0.411**	0.449	0.499	0.455	**0.409**	0.437	0.486	0.441	0.453	**0.408**	0.472	0.462	**0.386**	0.545	0.395	0.423
	IQR	0.157	0.179	**0.151**	0.242	**0.087**	0.106	0.164	0.189	0.143	0.182	**0.146**	0.171	0.202	0.106	0.089	0.121	**0.082**
ETTm2	Best	0.264	**0.255**	0.256	0.258	0.289	0.265	**0.255**	0.26	0.267	0.26	**0.253**	0.26	0.262	0.263	0.293	**0.253**	0.274
	Median	0.407	**0.34**	0.416	0.663	**0.32**	0.335	0.356	0.437	0.766	0.534	**0.326**	0.398	0.389	**0.34**	0.403	0.341	0.415
	IQR	0.565	**0.382**	0.854	0.9	**0.033**	0.911	0.441	0.711	1.086	0.906	**0.281**	0.836	0.86	0.836	**0.429**	0.922	0.753
ETTh1	Best	0.411	**0.402**	0.406	0.418	0.471	0.413	**0.401**	0.432	0.406	0.42	**0.401**	0.417	0.412	0.422	0.445	**0.401**	0.435
	Median	0.496	**0.48**	0.502	0.51	0.526	**0.475**	0.487	0.527	0.504	0.495	**0.481**	0.505	0.513	0.487	0.498	**0.482**	0.49
	IQR	0.115	**0.09**	0.167	0.19	**0.06**	0.07	0.105	0.207	0.214	0.149	**0.078**	0.137	0.244	0.128	**0.114**	0.156	0.114
ETTh2	Best	**0.327**	0.339	0.344	0.356	0.383	0.35	**0.325**	0.36	0.355	0.347	0.334	**0.329**	0.346	0.351	0.384	**0.325**	0.361
	Median	0.516	**0.427**	0.453	0.462	**0.41**	0.546	0.445	0.589	0.504	0.544	**0.421**	0.449	0.587	**0.422**	0.473	0.438	0.531
	IQR	0.642	**0.617**	1.453	2.021	**0.03**	0.754	0.65	1.345	1.393	2.209	**0.305**	1.086	1.483	**0.64**	0.842	1.394	0.721
ECL	Best	0.163	0.163	**0.157**	0.163	0.165	0.16	0.162	0.158	**0.157**	**0.158**	0.158	0.159	0.158	0.162	0.181	**0.157**	0.169
	Median	0.214	0.205	**0.201**	0.205	**0.181**	0.207	0.209	0.199	0.195	**0.194**	0.213	0.199	0.209	0.183	0.241	**0.182**	0.209
	IQR	0.056	0.06	**0.054**	0.054	**0.048**	0.055	0.06	0.048	0.052	0.052	0.06	**0.051**	0.054	**0.025**	0.044	0.047	0.041
traffic	Best	0.409	0.408	**0.394**	0.407	0.417	0.401	0.407	**0.394**	0.399	0.407	0.399	**0.394**	0.402	0.409	0.515	**0.394**	0.446
	Median	0.585	0.608	**0.558**	0.576	**0.475**	0.583	0.596	0.596	0.523	0.539	0.654	**0.507**	0.537	0.476	0.685	**0.453**	0.578
	IQR	**0.179**	0.215	0.195	0.208	**0.102**	0.181	0.198	0.199	0.19	0.149	**0.138**	0.195	0.163	0.135	**0.126**	0.181	0.144
weather	Best	0.222	**0.221**	0.221	0.227	**0.21**	0.229	0.22	0.226	0.221	0.227	**0.22**	0.222	0.223	0.227	0.253	**0.22**	0.239
	Median	**0.264**	0.268	0.266	0.279	**0.233**	0.264	0.265	0.274	0.25	0.272	**0.254**	0.27	0.282	0.248	0.286	**0.242**	0.258
	IQR	0.049	**0.047**	0.054	0.065	**0.018**	0.047	0.049	0.043	0.06	0.048	0.048	**0.044**	0.095	0.04	0.035	0.063	**0.031**
Exchange	Best	0.24	**0.238**	0.256	0.269	0.406	0.278	**0.237**	0.263	0.282	0.251	**0.238**	0.248	0.247	0.274	0.24	0.294	**0.238**
	Median	0.537	**0.435**	0.574	0.602	0.615	0.545	**0.473**	0.56	0.59	0.536	**0.443**	0.574	0.557	0.503	**0.401**	0.81	0.42
	IQR	0.445	**0.366**	0.517	0.499	**0.164**	0.6	0.433	0.492	0.492	0.604	**0.302**	0.56	0.506	0.349	**0.226**	0.822	0.245
ili	Best	1.619	1.561	**1.551**	1.597	1.665	1.672	**1.561**	1.637	1.642	1.603	1.629	1.63	**1.552**	1.878	1.715	2.269	**1.546**
	Median	2.953	2.851	**2.761**	2.731	**2.451**	2.949	2.889	2.791	2.728	**2.646**	3.043	2.815	2.755	2.622	2.705	3.799	**2.487**
	IQR	1.816	**1.516**	1.661	1.623	**0.656**	1.652	1.676	1.746	1.642	**1.55**	1.747	1.605	1.672	**1.069**	1.616	1.691	1.694
1st Count		3	**15**	9	0	**16**	2	7	1	1	4	**17**	5	1	6	5	**11**	5

(b) Part III – 7 Components (d_model, d_ff, etc.)

		d_model		d_ff		Encoder layers		Training Epochs			Loss Function			Learning Rate		Learning Rate Strategy	
dataset	stat	256	64	1024	256	2	3	10	20	50	HUBER	MAE	MSE	0.0001	0.001	null	type
ETTm1	Best	0.352	**0.35**	0.352	**0.35**	0.352	**0.35**	0.352	**0.351**	0.353	0.359	0.356	**0.35**	**0.35**	0.351	0.355	**0.35**
	Median	0.462	**0.423**	0.462	**0.423**	**0.424**	0.451	0.46	0.433	**0.428**	0.437	**0.433**	0.442	**0.425**	0.448	**0.431**	0.446
	IQR	0.166	**0.153**	0.166	**0.153**	**0.154**	0.156	0.173	0.163	**0.15**	**0.113**	0.139	0.166	**0.151**	0.175	**0.141**	0.176
ETTm2	Best	0.256	**0.253**	0.256	**0.253**	**0.254**	0.255	0.256	0.255	**0.254**	0.266	0.261	**0.253**	0.255	**0.253**	0.256	**0.253**
	Median	0.395	**0.35**	0.395	**0.35**	0.367	**0.363**	**0.352**	0.381	0.376	**0.353**	0.45	0.37	**0.359**	0.373	0.379	**0.355**
	IQR	0.792	**0.435**	0.792	**0.435**	0.785	**0.535**	**0.585**	0.767	0.607	**0.66**	0.681	0.665	**0.575**	0.729	**0.62**	0.783
ETTh1	Best	**0.401**	0.408	**0.401**	0.408	0.407	**0.401**	**0.402**	0.406	0.406	0.408	**0.401**	0.417	**0.401**	0.408	**0.402**	0.404
	Median	**0.491**	0.491	**0.491**	0.491	**0.49**	0.492	0.493	0.49	**0.489**	0.489	**0.487**	0.498	**0.477**	0.503	**0.49**	0.493
	IQR	**0.117**	0.134	**0.117**	0.134	**0.114**	0.132	0.119	0.156	**0.113**	0.137	0.125	**0.104**	**0.107**	0.134	**0.111**	0.141
ETTh2	Best	0.338	**0.326**	0.338	**0.326**	**0.325**	0.341	0.336	0.342	**0.325**	0.341	**0.326**	0.336	0.337	**0.325**	0.337	**0.325**
	Median	0.474	**0.441**	0.474	**0.441**	**0.442**	0.467	0.462	**0.442**	0.447	**0.415**	0.425	0.471	**0.446**	0.451	0.461	**0.445**
	IQR	**0.77**	0.871	**0.77**	0.871	0.902	**0.752**	1.286	0.806	**0.577**	**0.59**	0.928	0.901	**0.756**	0.956	0.969	**0.739**
ECL	Best	**0.157**	0.16	**0.157**	0.16	0.158	**0.157**	0.159	0.159	**0.157**	0.158	0.159	**0.157**	**0.157**	0.158	**0.157**	0.158
	Median	**0.204**	0.207	**0.204**	0.207	0.209	**0.202**	**0.205**	0.205	0.207	0.208	**0.193**	0.206	0.216	**0.199**	**0.198**	0.213
	IQR	0.057	**0.056**	0.057	**0.056**	0.058	**0.054**	0.057	**0.056**	0.057	0.055	**0.048**	0.057	0.062	**0.051**	**0.05**	0.06
traffic	Best	**0.394**	0.4	**0.394**	0.4	0.4	**0.394**	0.401	0.401	**0.394**	0.418	0.423	**0.394**	0.405	**0.394**	0.398	**0.394**
	Median	**0.548**	0.604	**0.548**	0.604	**0.566**	0.587	0.59	0.585	**0.562**	0.627	0.62	**0.57**	0.594	**0.56**	**0.551**	0.604
	IQR	**0.195**	0.206	**0.195**	0.206	**0.194**	0.197	0.209	0.194	**0.19**	**0.144**	0.164	0.195	0.216	**0.189**	**0.186**	0.212
weather	Best	**0.22**	0.22	**0.22**	0.22	**0.22**	0.22	0.224	**0.22**	0.22	0.223	0.225	**0.22**	0.222	**0.22**	0.221	**0.22**
	Median	0.27	**0.26**	0.27	**0.26**	0.266	**0.265**	0.268	**0.261**	0.267	0.277	**0.255**	0.266	**0.261**	0.27	**0.263**	0.268
	IQR	0.052	**0.049**	0.052	**0.049**	0.053	**0.05**	0.053	**0.048**	0.051	0.129	**0.039**	0.05	**0.044**	0.055	**0.046**	0.056
Exchange	Best	**0.237**	0.243	**0.237**	0.243	0.244	**0.239**	0.244	**0.239**	0.246	**0.241**	0.249	0.241	0.24	**0.238**	0.244	**0.238**
	Median	0.528	**0.465**	0.528	**0.465**	0.5	**0.494**	0.503	0.494	**0.489**	0.494	**0.442**	0.509	**0.444**	0.551	0.513	**0.486**
	IQR	0.518	**0.407**	0.518	**0.407**	0.495	**0.424**	0.488	**0.408**	0.451	0.449	**0.412**	0.468	**0.38**	0.545	0.495	**0.433**
ili	Best	**1.546**	1.632	**1.546**	1.632	1.564	**1.553**	1.586	**1.553**	1.613	**1.582**	1.59	1.585	1.662	**1.545**	1.591	**1.563**
	Median	**2.737**	2.971	**2.737**	2.971	**2.846**	2.854	2.9	**2.805**	2.85	2.9	2.933	**2.801**	3.161	**2.636**	**2.679**	3.203
	IQR	**1.599**	1.732	**1.599**	1.732	**1.635**	1.683	1.713	**1.611**	1.694	1.677	1.657	**1.653**	1.78	**1.47**	**1.438**	1.819
1st Count		**14**	13	**14**	13	12	**15**	4	11	**12**	8	**10**	9	**15**	12	**15**	12



(a) Series Sampling/Mixing (b) Sequence Length (c) Feature Attention

(d) *Hidden Layer Dimensions* (e) *FCN Layer Dimensions* (f) Encoder layers

(g) Loss Function (h) Learning Rate (i) Learning Rate Strategy

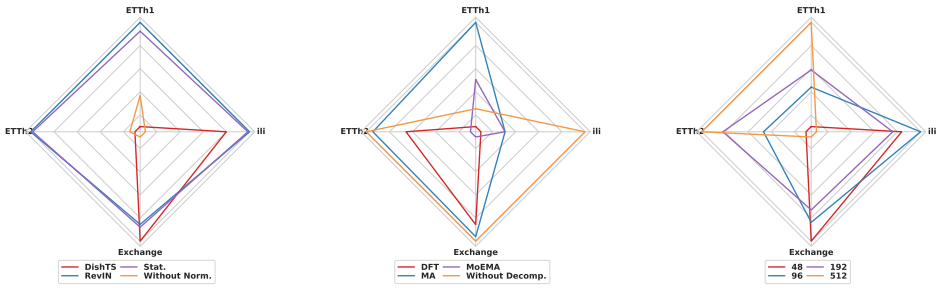
Figure H3: Overall performance across additional design dimensions in long-term forecasting. The results (MSE) are based on the top 25th percentile across all forecasting horizons.

H.2.1 Design Choices Evaluation Results for Long-term Forecasting Using MSE as the Metric

Spider Chart Analysis. Fig. H3 extends the baseline comparisons presented in Fig. 2 by employing multi-dimensional spider charts, where each vertex corresponds to a benchmark dataset. Closer proximity to the outer edge of a vertex indicates better performance of the associated design choice on that particular dataset. These visual representations offer an intuitive understanding of how different architectural decisions influence model effectiveness across diverse forecasting domains. Notably, configurations for components including Series Sampling/Mixing (Fig. H3a), Hidden Layer Dimensions (Fig. H3d), FCN Layer Dimensions (Fig. H3e), Learning Rate (Fig. H3h), and Learning Rate Strategy (Fig. H3i) demonstrate similar spatial patterns in the radar charts. Specifically, ECL, ILI, and Traffic datasets exhibit consistent parameter preferences across these components, suggesting intrinsic alignment between their temporal patterns and specific architectural configurations.

In addition, Fig. H4 provides a broader evaluation of large-scale time series models, revealing that conventional architectures still maintain a competitive advantage over LLM-based models, especially in domain-specific forecasting tasks where structural inductive biases play a crucial role.

Box Plots Analysis. The impact of various design choices for each architectural component is further illustrated through box plots in Fig. H5 and Fig. H6. These visualizations complement the spider charts by providing a statistical perspective on performance variability and robustness across multiple benchmark datasets. Together, the two forms of analysis offer a comprehensive view of how different configurations affect forecasting accuracy.

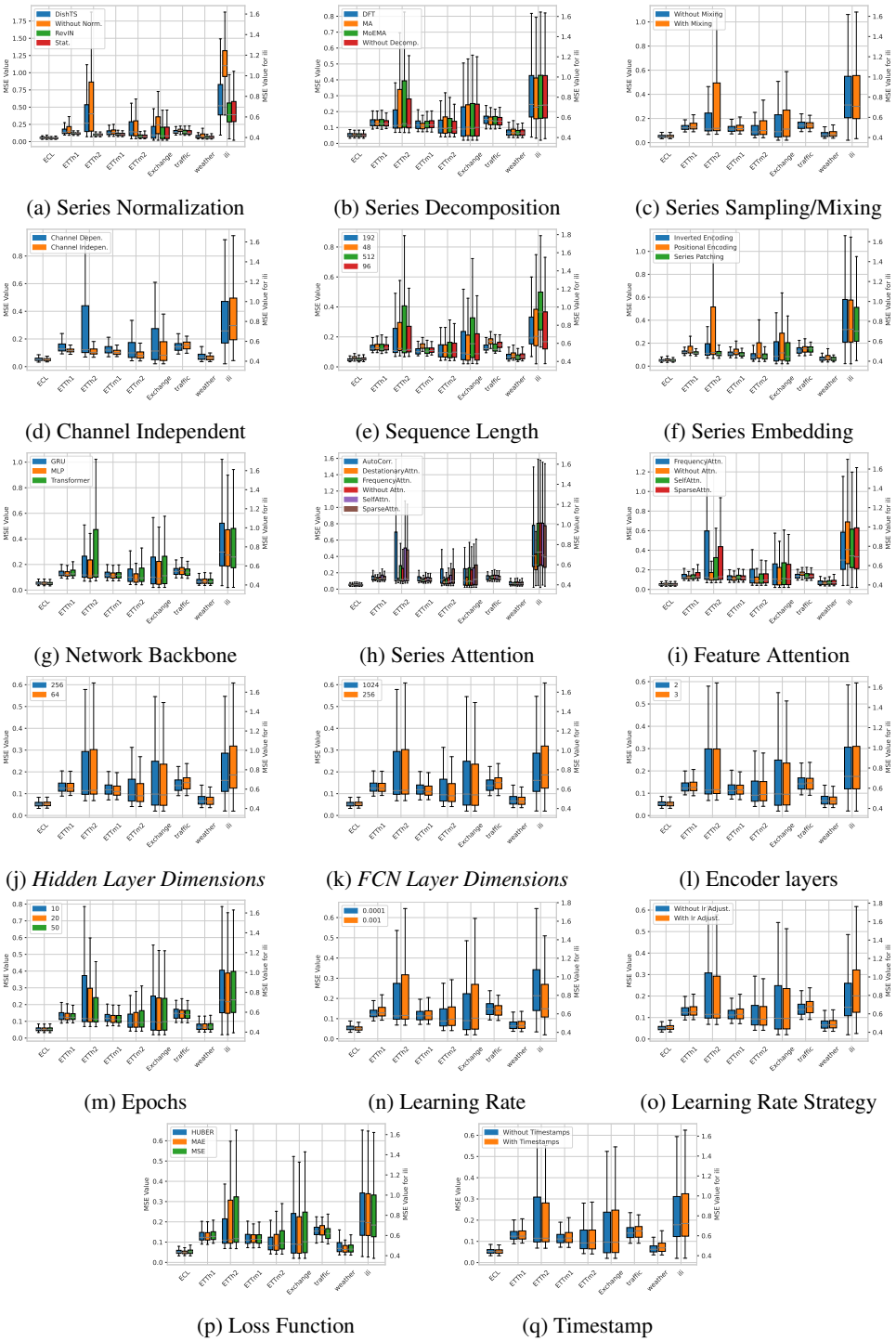


(a) Series Normalization (b) Series Decomposition (c) Sequence Length

(d) Network Backbone (e) *Hidden Layer Dimensions* (f) *FCN Layer Dimensions*

(g) Epochs (h) Learning Rate (i) Learning Rate Strategy

Figure H4: Overall performance across all design dimensions when using LLMs or TSFMs in long-term forecasting. The results (MSE) are based on the top 25th percentile across all forecasting horizons.



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Sequence Length (f) Series Embedding

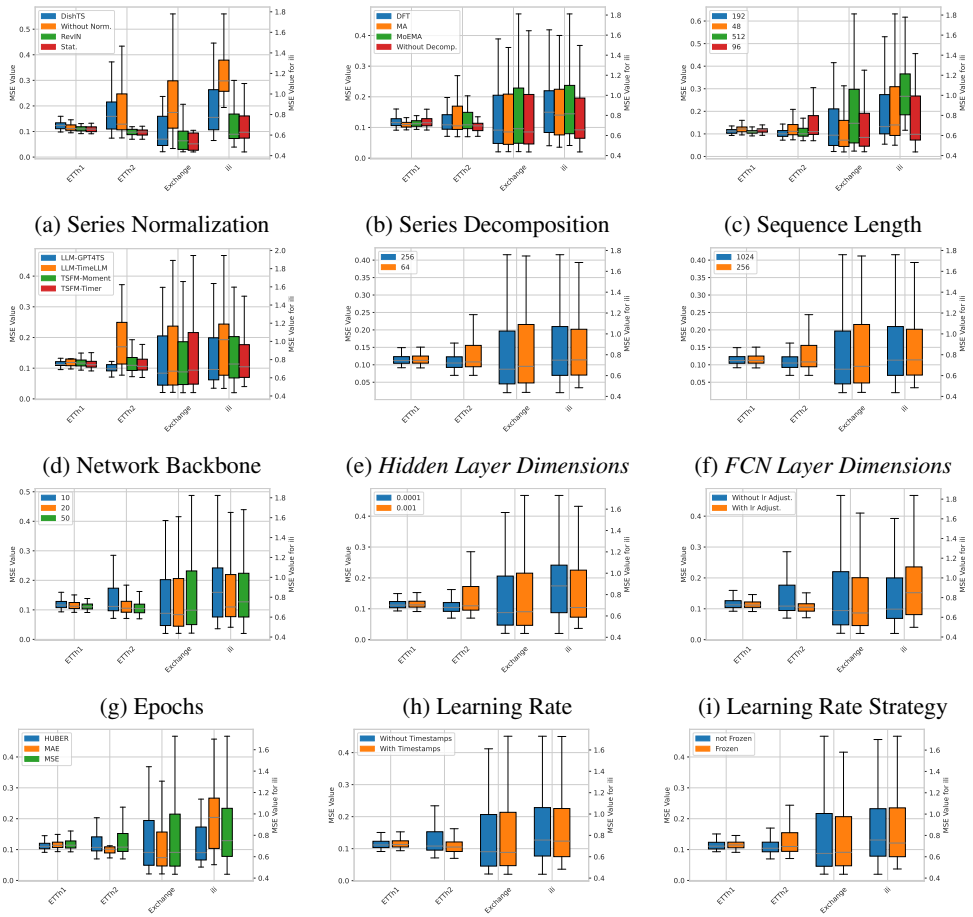
(g) Network Backbone (h) Series Attention (i) Feature Attention

(j) *Hidden Layer Dimensions* (k) *FCN Layer Dimensions* (l) Encoder layers

(m) Epochs (n) Learning Rate (o) Learning Rate Strategy

(p) Loss Function (q) Timestamp

Figure H5: Overall performance across all design dimensions in long-term forecasting. The results (**MSE**) are averaged across all forecasting horizons. Due to the significantly different value range and variability of the ILI dataset compared to other datasets, its box plot is plotted using the right-hand y-axis, while all other datasets share the left-hand y-axis.



(a) Series Normalization (b) Series Decomposition (c) Sequence Length

(d) Network Backbone (e) *Hidden Layer Dimensions* (f) *FCN Layer Dimensions*

(g) Epochs (h) Learning Rate (i) Learning Rate Strategy

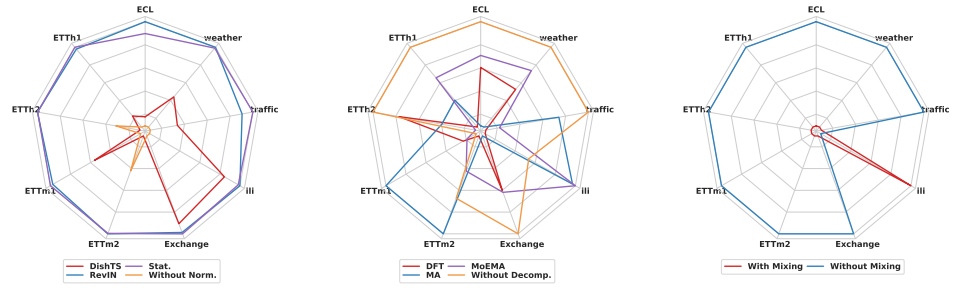
(j) Loss Function (k) Timestamp (l) Frozen

Figure H6: Overall performance across all design dimensions when using LLMs or TSFMs in long-term forecasting. The results (**MSE**) are averaged across all forecasting horizons. Due to the significantly different value range and variability of the ILI dataset compared to other datasets, its box plot is plotted using the right-hand y-axis, while all other datasets share the left-hand y-axis.

H.2.2 Design Choices Evaluation Results for Long-term Forecasting Using MAE as the Metric

For the MAE-based performance evaluation, we analyze the effects of different design choices using both spider charts and box plots (Fig. H7 and Fig. H8). These visualizations complement the MSE-based analysis and confirm the generalizability of our findings across error metrics. In particular, normalization methods such as RevIN and Stationary consistently achieve the lowest MAE values, underscoring their effectiveness in mitigating non-stationarity. Similarly, decomposition strategies exhibit selective benefits: MA-based methods improve predictions on datasets like ETTh1 and ETTm2, while raw-series modeling remains more effective on ECL and Traffic, where decomposition tends to degrade performance.

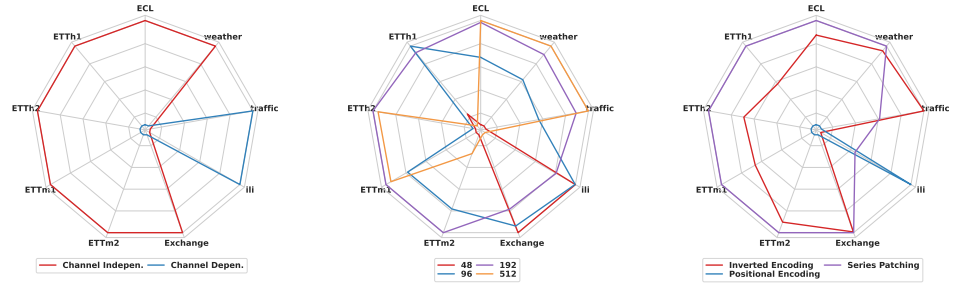
Beyond preprocessing, MAE evaluations further validate the consistency of our architectural insights. Channel-independent designs retain strong performance across most datasets, except on Traffic and ILI, where localized dependencies dominate. Tokenization methods show stable ranking across both metrics, with patch-wise encoding consistently outperforming point-wise approaches. Notably, complex architectures such as Transformers provide only marginal gains over MLPs in certain cases (e.g., Traffic), suggesting that their benefits may not justify the added complexity. Overall, the alignment between MAE and MSE results reinforces the robustness of our design principles, demonstrating that the observed patterns are not metric-specific but instead reflect core relationships between architecture and forecasting performance.



(a) Series Normalization

(b) Series Decomposition

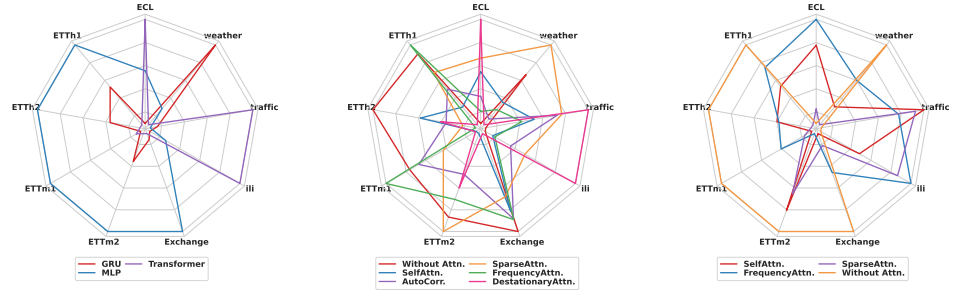
(c) Series Sampling/Mixing



(d) Channel Independent

(e) Sequence Length

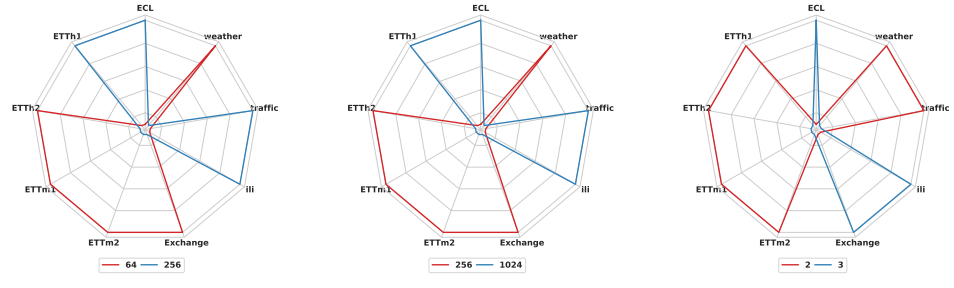
(f) Series Embedding



(g) Network Backbone

(h) Series Attention

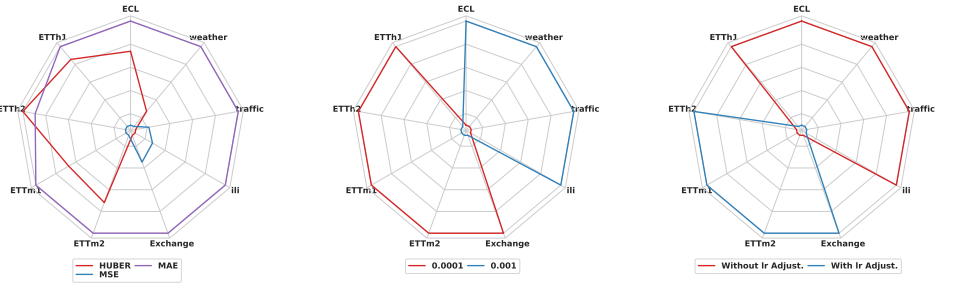
(i) Feature Attention



(j) *Hidden Layer Dimensions*

(k) *FCN Layer Dimensions*

(l) Encoder layers

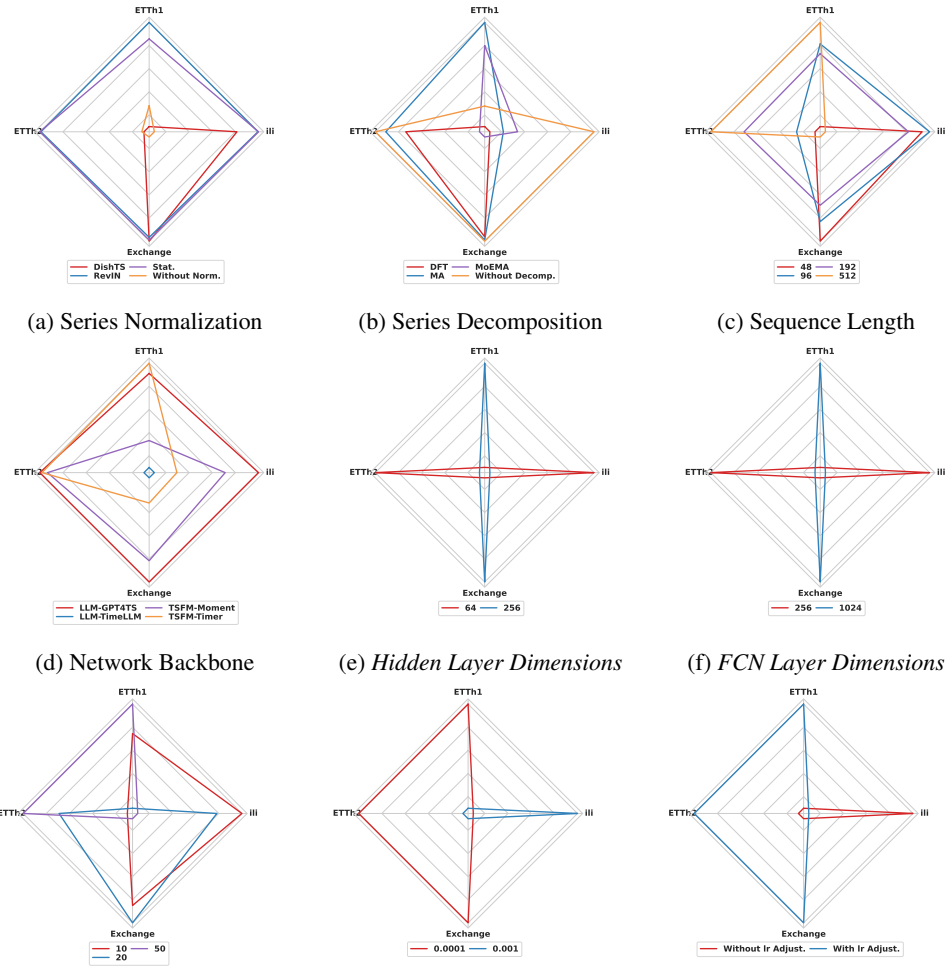


(m) Loss Funtion

(n) Learning Rate

(o) Learning Rate Strategy

Figure H7: Overall performance across key design dimensions in long-term forecasting. The results (**MAE**) are based on the top 25th percentile across all forecasting horizons.

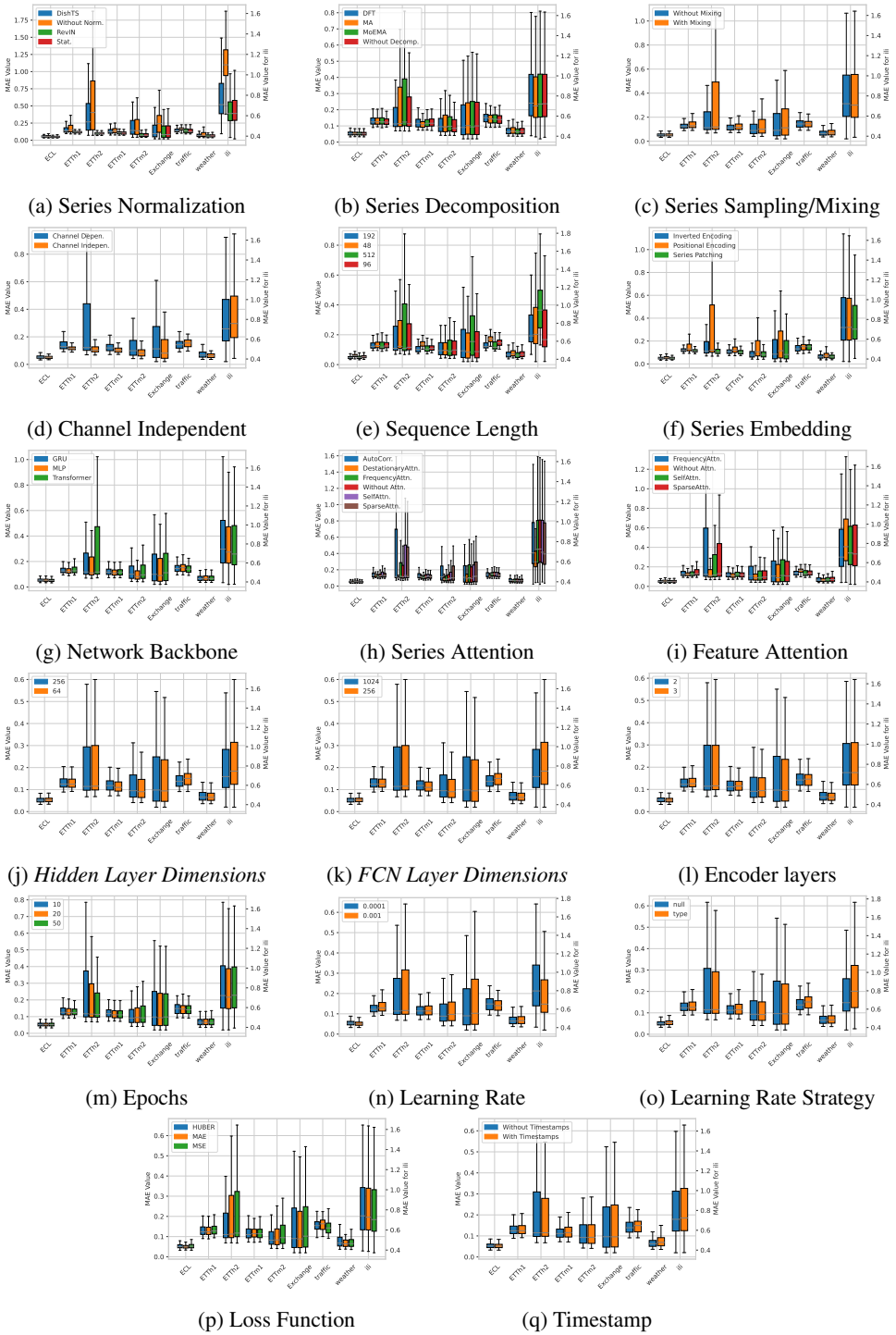


(a) Series Normalization (b) Series Decomposition (c) Sequence Length

(d) Network Backbone (e) *Hidden Layer Dimensions* (f) *FCN Layer Dimensions*

(g) Epochs (h) Learning Rate (i) Learning Rate Strategy

Figure H8: Overall performance across all design dimensions when using LLMs or TSFMs in long-term forecasting. The results (**MAE**) are based on the top 25th percentile across all forecasting horizons.



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Sequence Length (f) Series Embedding

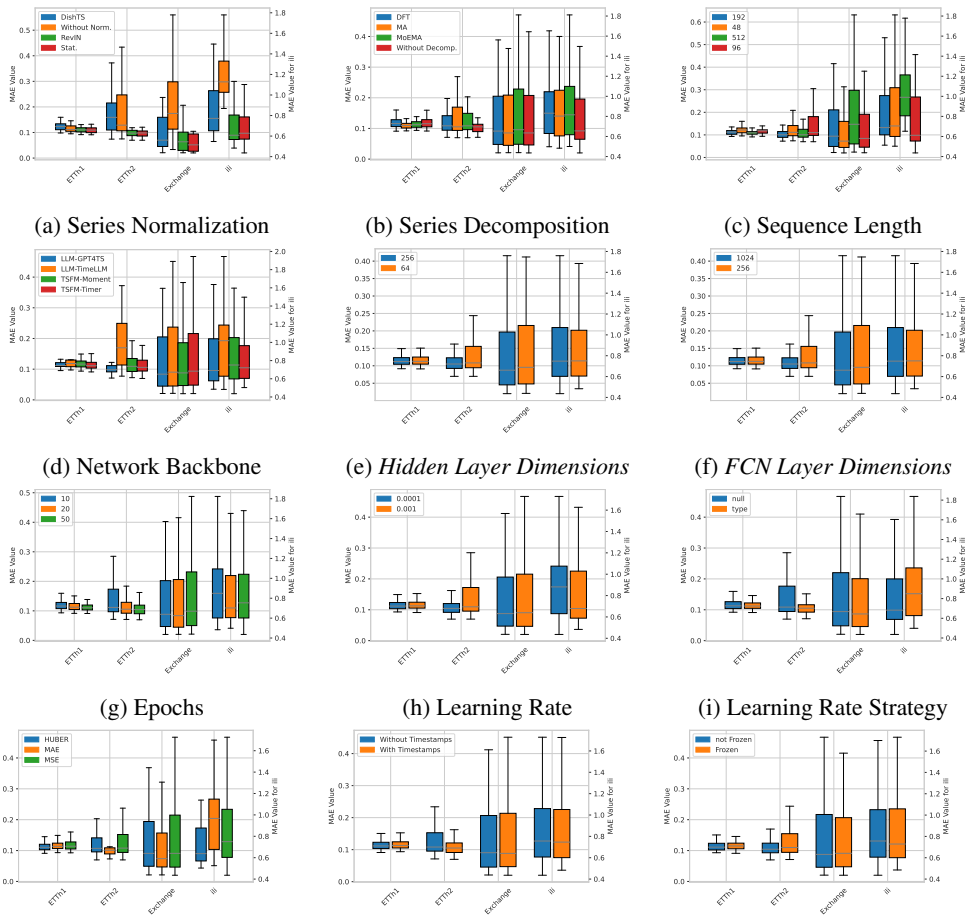
(g) Network Backbone (h) Series Attention (i) Feature Attention

(j) *Hidden Layer Dimensions* (k) *FCN Layer Dimensions* (l) Encoder layers

(m) Epochs (n) Learning Rate (o) Learning Rate Strategy

(p) Loss Function (q) Timestamp

Figure H9: Overall performance across all design dimensions in long-term forecasting. The results (**MAE**) are averaged across all forecasting horizons.



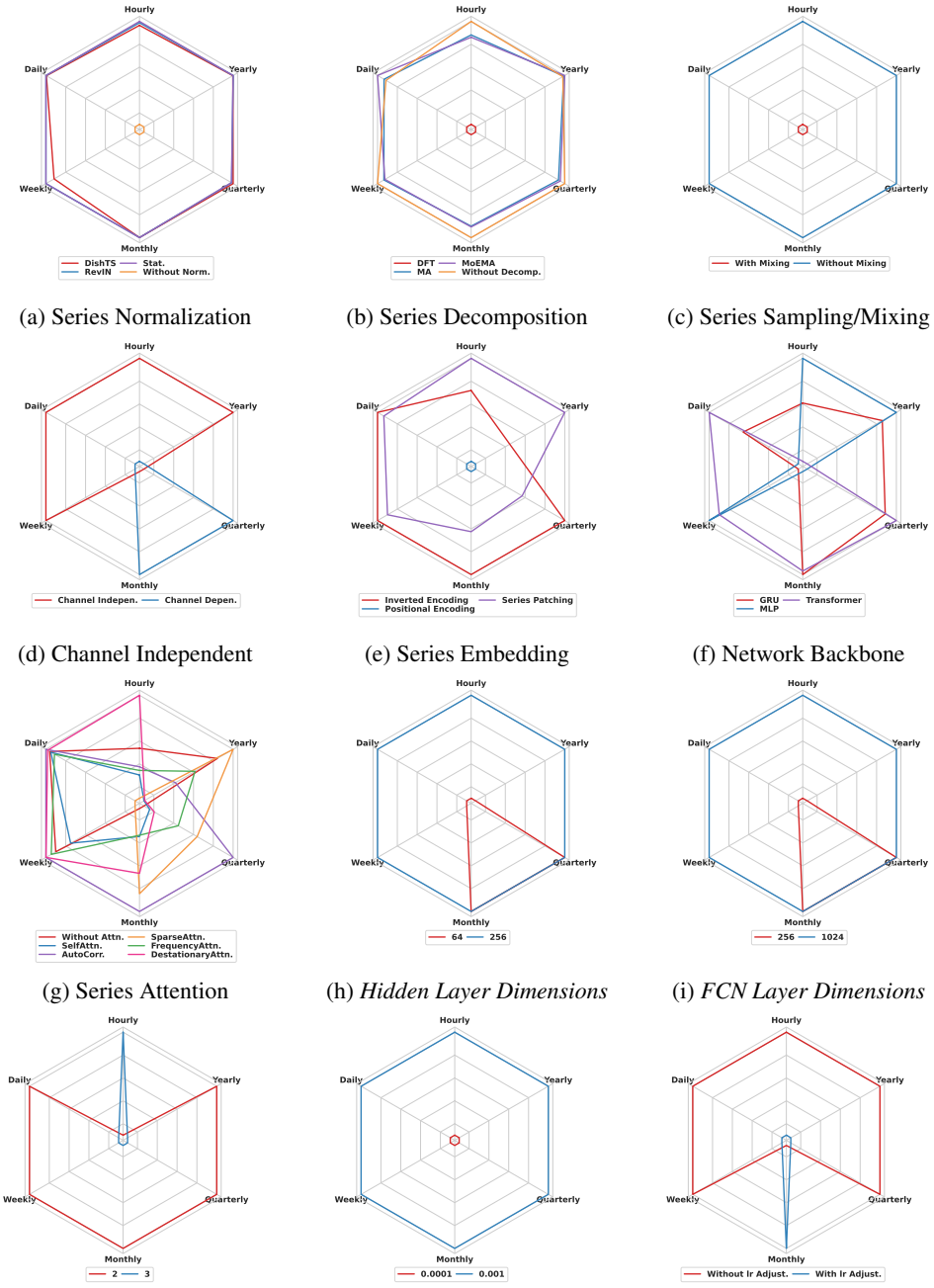
(a) Series Normalization (b) Series Decomposition (c) Sequence Length

(d) Network Backbone (e) *Hidden Layer Dimensions* (f) *FCN Layer Dimensions*

(g) Epochs (h) Learning Rate (i) Learning Rate Strategy

(j) Loss Function (k) Timestamp (l) Frozen

Figure H10: Overall performance across all design dimensions when using LLMs or TSFMs in long-term forecasting. The results (**MAE**) are averaged across all forecasting horizons.



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Series Embedding (f) Network Backbone

(g) Series Attention (h) *Hidden Layer Dimensions* (i) *FCN Layer Dimensions*

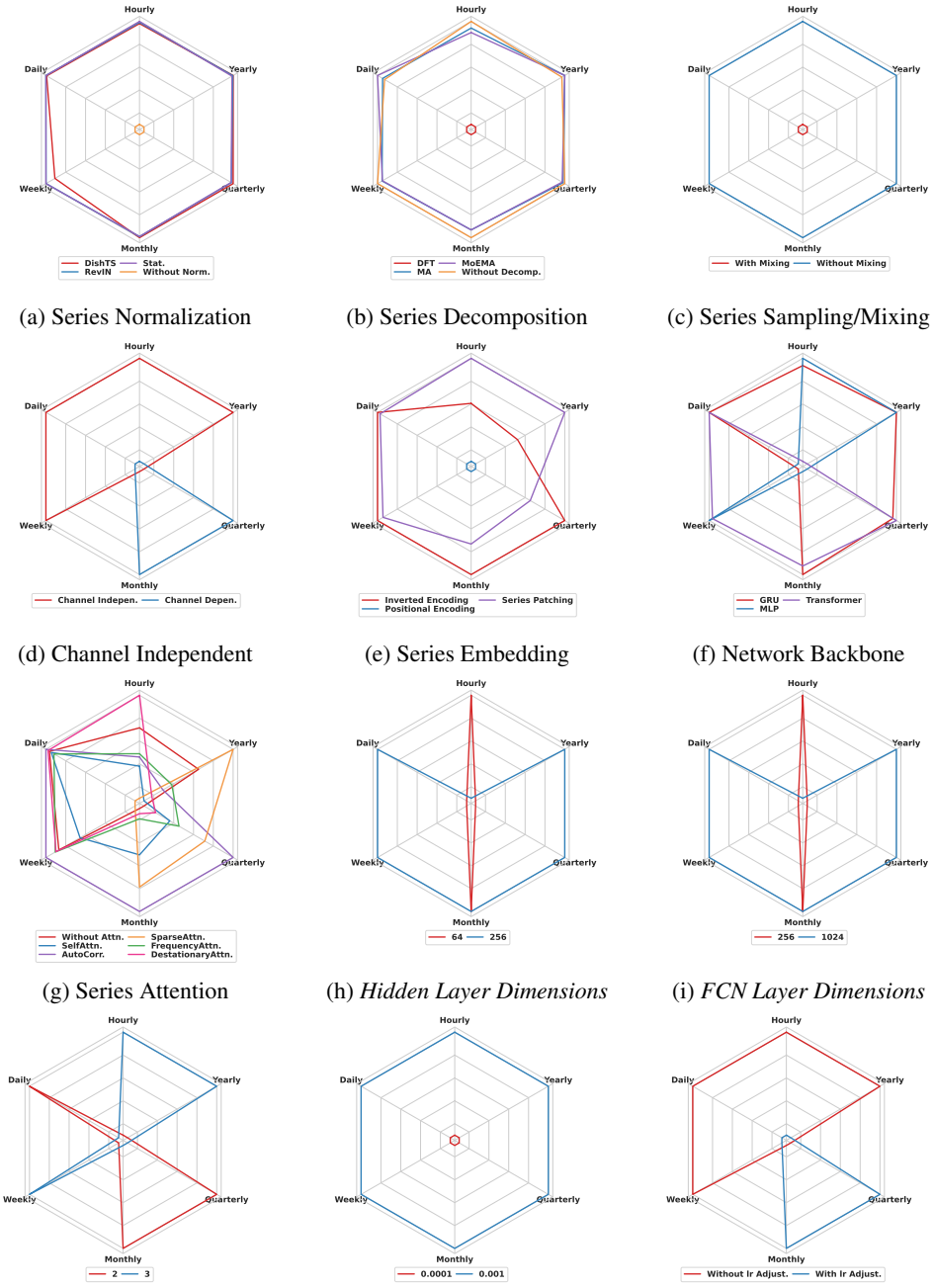
(j) Encoder layers (k) Learning Rate (l) Learning Rate Strategy

Figure H11: Overall performance across all design dimensions in short-term forecasting. The results (**MASE**) are based on the top 25th percentile across all forecasting horizons.

H.3 Complete Evaluation Results of Short-term Forecasting Using MASE, OWA and sMAPE as the Metric

For short-term forecasting, we comprehensively evaluate different design dimensions using both spider charts and box plots. The spider charts—shown in Figure H11, Figure H12, and Figure H13—visualize performance across datasets, with each vertex representing a benchmark dataset. Closer proximity to a vertex indicates stronger performance of a particular design choice in that dataset.

Complementary box plots are provided in Figure H14, Figure H15, and Figure H16, offering a statistical perspective on the distribution and robustness of performance across evaluation metrics.



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Series Embedding (f) Network Backbone

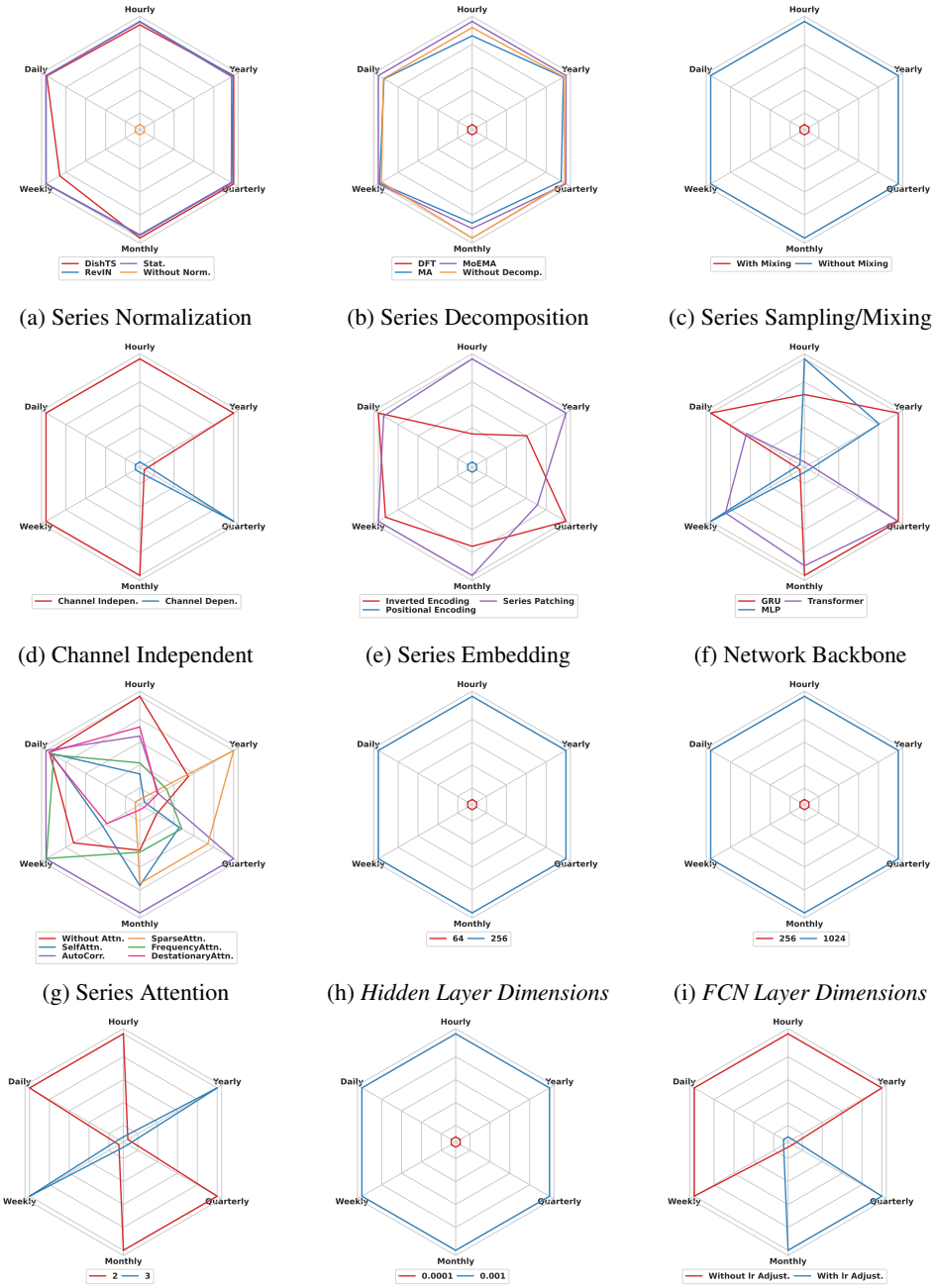
(g) Series Attention (h) *Hidden Layer Dimensions* (i) *FCN Layer Dimensions*

(j) Encoder layers (k) Learning Rate (l) Learning Rate Strategy

Figure H12: Overall performance across all design dimensions in short-term forecasting. The results (**OWA**) are based on the top 25th percentile across all forecasting horizons.

Overall, the relative performance trends observed under MASE, OWA, and sMAPE metrics are consistent with those found in long-term forecasting tasks, reinforcing the generalizability and stability of our architectural choices.

H.4 Ablation-based experiment for investigating four phases.



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Series Embedding (f) Network Backbone

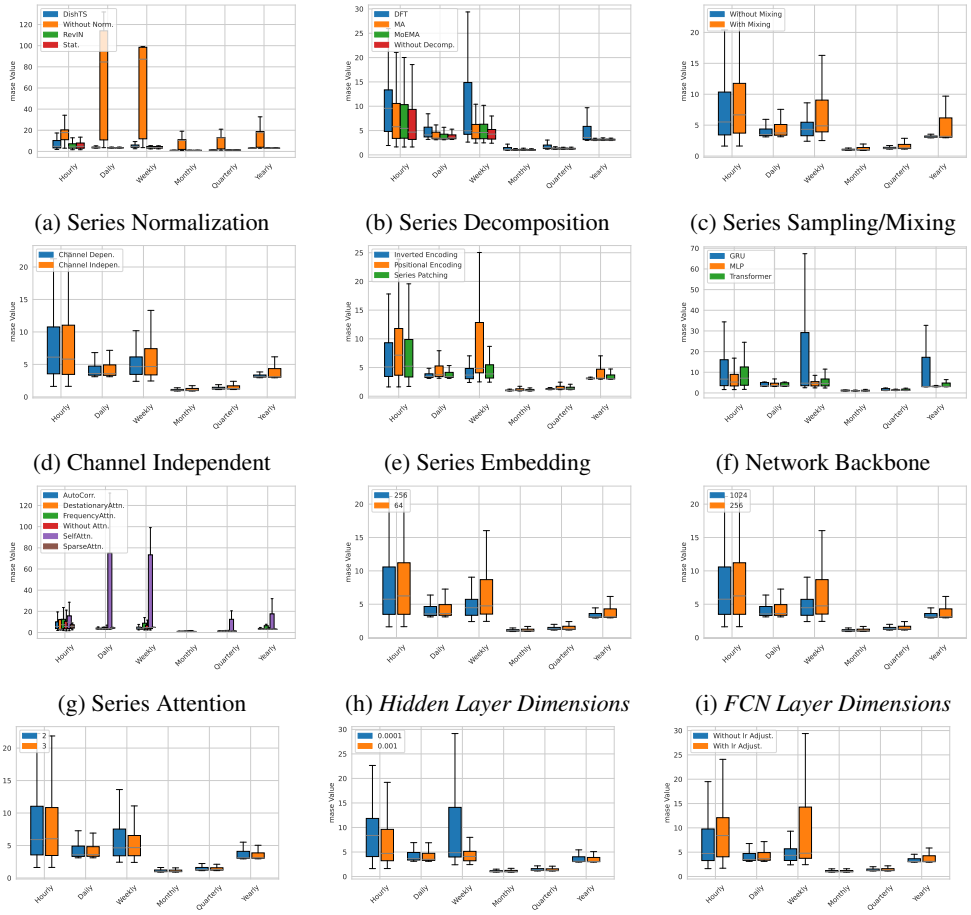
(g) Series Attention (h) *Hidden Layer Dimensions* (i) *FCN Layer Dimensions*

(j) Encoder layers (k) Learning Rate (l) Learning Rate Strategy

Figure H13: Overall performance across all design dimensions in short-term forecasting. The results (**SMAPE**) are based on the top 25th percentile across all forecasting horizons.

To quantitatively investigate the influence of four phases during MTSF pipeline (shown in Fig. 1)—Series Preprocessing, Series Encoding, Network Architecture, and Network Optimization, we design an ablation-based experiment. We find that, compared to complex model structures and optimization methods, the series preprocessing and encoding phases are more crucial.

Firstly, we establish the typical design choices as a baseline for 9 original datasets and 3 additional datasets (AQShunyi, AQWan, and CzeLan), which is defined as follows: *Series Normalization=Null, Series Decomposition=Null, Series Sampling=False, Channel Independent=False, Sequence Length=48, Series Embedding=Positional Encoding, Network Backbone=RNN, Series Attention=Null, Feature Attention=Null,*



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Series Embedding (f) Network Backbone

(g) Series Attention (h) *Hidden Layer Dimensions* (i) *FCN Layer Dimensions*

(j) Encoder layers (k) Learning Rate (l) Learning Rate Strategy

Figure H14: Overall performance across all design dimensions in short-term forecasting. The results are based on **MASE**.

Encoder Layers=2, Epochs=10, Loss Function=MSE, Learning Rate=1e-3, Learning Rate Strategy=Null. **This baseline is applied to 48 experimental settings across 12 datasets and 4 different forecasting lengths.**

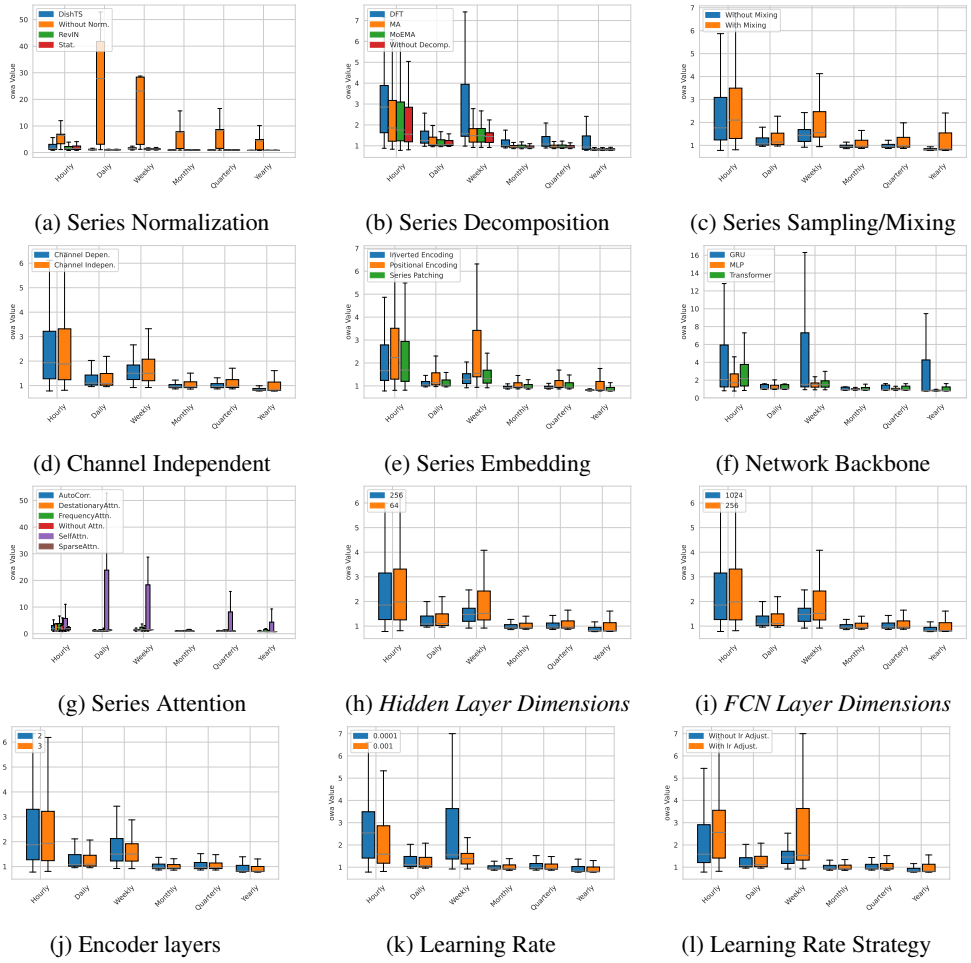
Next, to evaluate the impact of each phase, we keep the other three phases fixed at their simplest design choices and select a range of combinations for the current phase under evaluation. For example, when evaluating the impact of the Network Architecture phase, we fix the other three phases to the combinations *Series Normalization=Null, Series Decomposition=Null, Series Sampling=False, Channel Independent=False, Series Embedding=Positional Encoding, Loss Function=MSE, Learning Rate=1e-3, Learning Rate Strategy=Null* and isolate the results varied during this phase.

To account for differences across datasets, we measured the influence of each phase by comparing the relative improvement of the current phase against the corresponding baseline for each dataset and forecasting length. This is formally defined as:

$$AvgRelErr_{phase} = \frac{1}{|S_{phase}|} \sum_{s \in S_{phase}} \left(-\frac{MSE_s^{variant} - MSE_s^{baseline}}{MSE_s^{baseline}}\right)$$

where S_{phase} is the set of all (dataset, horizon) settings that satisfy the ablation criteria for the phase; $MSE_s^{variant}$ is the MSE of a candidate configuration under setting s; $MSE_s^{baseline}$ is the MSE of the minimal baseline under the same setting s.

The table H13 reports the maximum, mean, and standard deviation of the relative improvement for each phase compared to the corresponding baseline. These values represent the potential, average improvement



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Series Embedding (f) Network Backbone

(g) Series Attention (h) *Hidden Layer Dimensions* (i) *FCN Layer Dimensions*

(j) Encoder layers (k) Learning Rate (l) Learning Rate Strategy

Figure H15: Overall performance across all design dimensions in short-term forecasting. The results are based on **OWA**.

Table H13: **Performance Improvement of Each Phrase over the Baseline.**

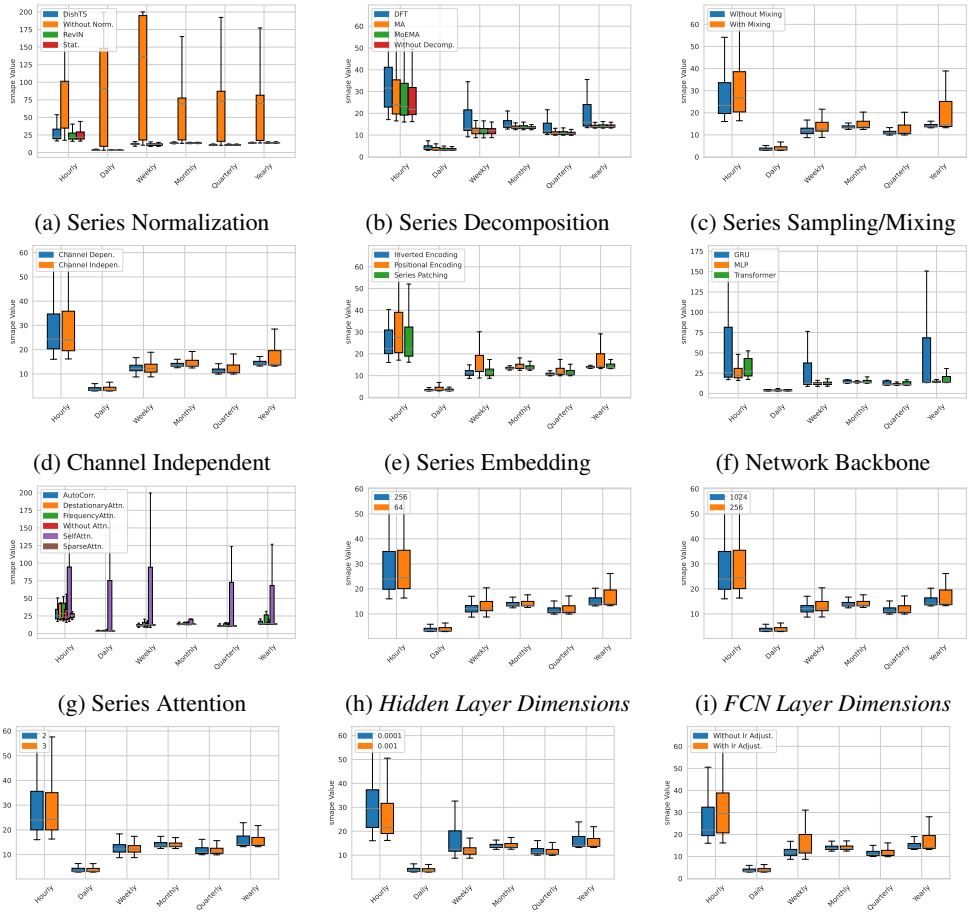
Phase	Max Improvement	Mean Improvement	Std Improvement
Series Preprocessing	90.87%	19.96%	0.394
Series Encoding	83.90%	30.98%	0.248
Network Architecture	66.75%	7.12%	0.290
Network Optimization	26.54%	10.01%	0.109

level, and stability of the current phase, respectively. In summary, compared to complex model structures and optimization methods, the series preprocessing and encoding phases are more crucial.

H.5 Explaining Design Drivers via Meta-Feature Importance Analysis

To directly investigate the impact of individual meta-features, we conducted an additional analysis using an interpretable XGBoost-based meta-learner. Although this machine learning–based variant slightly underperforms compared to the original deep learning–based meta-learner (average MAE 0.447 vs. 0.426), due to its limited capacity in modeling rich, high-dimensional interactions among features, it remains competitive and provides a clear advantage in interpretability.

This analysis allows us to quantify the relative importance of meta-features and structural design dimensions in determining model performance. As summarized in Table H14, certain temporal and spectral features—such as MFCC descriptors and Negative Turning Points—consistently appear among the most



(a) Series Normalization (b) Series Decomposition (c) Series Sampling/Mixing

(d) Channel Independent (e) Series Embedding (f) Network Backbone

(g) Series Attention (h) *Hidden Layer Dimensions* (i) *FCN Layer Dimensions*

(j) Encoder layers (k) Learning Rate (l) Learning Rate Strategy

Figure H16: Overall performance across all design dimensions in short-term forecasting. The results are based on **SMAPE**.

Table H14: **Top 5 Most Important Meta-Features per Dataset Estimated via XGBoost**

Dataset	Top 5 Meta-Features (Importance)				
ETTm1	mean_Negativeturningpoints (0.08)	series norm (0.06)	mean_Centroid (0.05)	mean_MFCC_0 (0.05)	min_Positiveturningpoints (0.05)
ETTm2	mean_Negativeturningpoints (0.10)	series norm (0.05)	mean_MFCC (0.05)	mean_Centroid (0.05)	q25_Kurtosis (0.04)
ETTh1	mean_MFCC_10 (0.08)	series norm (0.06)	mean_Spectralroll-on (0.05)	mean_Spectraldistance (0.04)	mean_MFCC (0.04)
ETTh2	mean_MFCC_0 (0.07)	series norm (0.05)	mean_Medianfrequency (0.05)	mean_Centroid (0.04)	min_Meanabsolutediff (0.04)
ECL	mean_MFCC (0.08)	std_MFCC (0.08)	series norm (0.07)	mean_Centroid (0.06)	mean_Maxpowerspectrum (0.04)
Traffic	q25_Kurtosis (0.08)	series norm (0.07)	mean_Centroid (0.06)	mean_Maxpowerspectrum (0.05)	mean_MFCC (0.05)
Weather	mean_MFCC (0.14)	mean_Negativeturningpoints (0.11)	series norm (0.06)	min_Negativeturningpoints (0.05)	mean_Centroid (0.04)
Exchange	std_MFCC (0.11)	mean_Negativeturningpoints (0.10)	mean_MFCC (0.07)	series norm (0.06)	mean_Medianfrequency (0.03)
ILI	mean_MFCC (0.09)	mean_Maximumfrequency (0.06)	channel independent (0.05)	series norm (0.05)	mean_LPCC (0.05)

influential across datasets. In addition, architectural design choices like series normalization emerge as universally important factors, further validating the findings of our component-level ablation study.

H.6 Meta-Feature Similarity Enables Targeted Knowledge Transfer

In Fig. G2, we visualize the dimension-reduced meta-features across different datasets using PCA. The visualization confirms that datasets tend to cluster based on inherent properties, such as domain (e.g., ETT family) and temporal frequency (e.g., M4-Hourly vs. M4-Yearly). This indicates that meta-feature similarity reflects structural characteristics of datasets, and suggests the potential for targeted knowledge transfer between similar datasets.

To further explore this, we conducted a case study focusing on the ILI dataset—a relatively difficult and data-scarce task. We enriched the meta-learner's training pool by adding two datasets (COVID-19 and FRED-MD) that are more similar to ILI in the meta-feature space. As shown in Table H15, TSGym's

performance on ILI improves significantly, while performance on other datasets remains stable or even improves slightly. This highlights the potential of incorporating similar datasets to enhance performance on low-resource or underperforming tasks.

Table H15: **Performance Comparison Before and After Adding Similar Datasets (COVID-19 and FRED-MD) to the Meta-Learner Training Pool**

Metric	TSGym MSE	TSGym MAE	+COVID-19, FRED-MD MSE	+COVID-19, FRED-MD MAE
ETTm1	**0.357**	**0.383**	0.360	0.386
ETTm2	0.261	**0.319**	**0.258**	0.320
ETTh1	**0.426**	0.440	0.429	**0.436**
ETTh2	**0.358**	**0.400**	0.368	0.406
ECL	0.170	**0.265**	0.170	0.269
Traffic	0.435	0.313	**0.428**	**0.306**
Weather	0.229	0.268	**0.228**	**0.266**
Exchange	0.410	0.431	**0.373**	**0.408**
ILI	2.233	1.015	**2.098**	**0.941**

H.7 Meta-Learner Performance Scaling with Candidate Pool Size

To investigate how the size of the candidate model pool M_s affects meta-learner performance, we conducted a scaling analysis across all datasets. We trained the meta-learner on progressively larger subsets of M_s (ranging from 5% to 100%), and measured the average rank of the model selected by TSGym.

As shown in Table H16, the performance improves significantly as the pool size increases up to 25%, after which the gains plateau. Remarkably, even with just 10% of the full pool, TSGym already outperforms strong baselines such as DUET (which achieves average ranks of 4.11 for MSE and 3.67 for MAE). This highlights the high sample efficiency of TSGym and suggests that a moderately sized pool is sufficient to reach near-optimal performance. These results further motivate the use of smarter sampling strategies, such as Bayesian Optimization, to construct high-quality training pools with minimal cost.

Table H16: **Effect of Candidate Pool Size on Meta-Learner Selection Accuracy**

Subset Size of M_s	MSE (Avg. Rank)	MAE (Avg. Rank)
5%	3.67	3.44
10%	2.67	3.00
25%	1.67	1.78
50%	1.89	2.67
75%	1.89	2.22
100%	1.67	2.00

I Comparative Experiments with AutoML Methods

I.1 Experimental Setup

We investigated a number of well-known AutoML and NAS (Neural Architecture Search) libraries, including TPOT[47], H2O-3[22], Microsoft NNI[45], Auto-Keras[25], Auto-Sklearn[18], and NASLib[51]. We noted, however, that most of these frameworks are not specifically designed for time-series forecasting, which would make adapting them for MTSF tasks both burdensome and likely result in an unfair comparison. For this reason, we selected two leading AutoML libraries that explicitly support MTSF for a direct and meaningful comparison: AutoGluon-TimeSeries[55] and AutoTS[10].

The experimental settings for AutoGluon and AutoTS were aligned with those of TSGym. We adopted the same training, validation, and test set splits as used in TSGym, with all base configurations set to their default values. To ensure computational efficiency, AutoGluon was configured with the "high_quality" preset, while AutoTS was tested using the "superfast" model setting.

Table I17: Short-term Forecasting Comparison

Model	TSGym (ours)	AutoGluon	AutoTS
OWA	**0.872**	0.95	2.002
SMAPE	**12.013**	13.178	18.977
MASE	**1.575**	1.775	4.981

Table I18: Long-term Forecasting Comparison

Dataset	**TSGym (ours)**		**AutoGluon**		**AutoTS**	
	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.357**	**0.383**	0.482	0.408	0.744	0.546
ETTm2	**0.261**	**0.319**	0.273	0.337	0.392	0.389
ETTh1	**0.426**	**0.440**	0.503	0.473	0.981	0.610
ETTh2	**0.358**	**0.400**	0.419	0.430	0.589	0.488
ECL	**0.170**	**0.265**	0.265	0.328	0.327	0.355
Traffic	**0.435**	0.313	0.555	0.325	0.739	**0.311**
Weather	**0.229**	**0.268**	0.236	0.270	0.519	0.372
Exchange	0.410	0.431	**0.330**	**0.393**	0.588	0.494
ILI	**2.233**	1.015	2.271	**0.979**	2.533	1.049

I.2 Performance Comparison

We benchmarked TSGym against these two methods on both short-term and long-term forecasting tasks. The results are presented in Table I17 and Table I18, respectively.

For short-term forecasting (Table I17), TSGym demonstrates clear superiority, achieving the best scores across all three metrics: Overall Weighted Average (OWA), Symmetric Mean Absolute Percentage Error (SMAPE), and Mean Absolute Scaled Error (MASE). This indicates a robust and consistently better performance in short-horizon predictions compared to the established AutoML baselines.

In the more challenging long-term forecasting tasks (Table I18), TSGym continues to show a strong competitive advantage. It secures the lowest (best) Mean Squared Error (MSE) and Mean Absolute Error (MAE) on the majority of datasets, including ETTm1, ETTm2, ETTh1, ETTh2, ECL, and Weather. It is worth noting that AutoGluon achieves better performance on the Exchange dataset and a lower MAE on the ILI dataset, while AutoTS shows a competitive MAE on the Traffic dataset. Nevertheless, TSGym's dominant performance across a wide range of datasets underscores its effectiveness and robustness for long-horizon prediction.

I.3 Conclusion

As the results demonstrate, TSGym consistently and significantly outperforms both AutoGluon and AutoTS across the vast majority of datasets for both short-term and long-term forecasting tasks.

In summary, while TSGym shares the goals of AutoML, its distinct methodology—automated model construction via fine-grained component decomposition and meta-learning—offers a clear advantage over existing AutoML methods for MTSF. This specialized design, now validated against both state-of-the-art deep-learning MTSF methods and the newly included AutoML baselines, represents a significant and valuable contribution to the MTSF community.

J TSGym Performance Comparison Across Sampling Strategies

To reduce computational cost and improve search efficiency, we adopted Optuna, which is based on Bayesian optimization, as a more efficient sampling strategy than random search. After an initial 50 random trials, Optuna intelligently sampled an additional 50 configurations. The experimental results confirms that Optuna significantly improves search efficiency, discovering superior model combinations with far fewer evaluations.

Table J19 reports the MSE distribution statistics for configurations sampled by Optuna and random search across various datasets. "Trials to Exceed Best Random" indicates how many Optuna trials were needed to outperform the best result from random search. Optuna consistently achieves better results with significantly fewer evaluations (less than 100 trials vs. 1.5-5 times more for random search), validating its sampling efficiency. This confirms that a meta-learner trained on this intelligently curated set is more effective.

Table J19: **Comparison of MSE Distribution Between Optuna and Random Search Across Datasets**

Dataset	Method	Optuna Best Over Random Experiments	Min mse	Q1 mse	Median mse	Q3 mse	Max mse	Total Experiment Count
ECL	Optuna	56	**0.131**	**0.158**	**0.182**	**0.213**	**0.414**	**400**
	Random		0.134	0.179	0.212	0.247	0.862	1493
ETTh1	Optuna	70	**0.355**	**0.416**	**0.450**	**0.519**	**1.210**	**400**
	Random		0.376	0.442	0.494	0.581	1.860	1129
ETTh2	Optuna	91.5	**0.268**	**0.342**	**0.400**	**0.539**	**9.060**	**400**
	Random		0.270	0.391	0.464	1.017	30.771	1147
ETTm1	Optuna	59.5	**0.293**	**0.341**	**0.405**	**0.472**	**4317.839**	**400**
	Random		0.295	0.378	0.447	0.547	304538.500	623
ETTm2	Optuna	56.5	**0.159**	**0.221**	**0.279**	**0.385**	**9.344**	**400**
	Random		0.167	0.259	0.371	0.584	198.023	762
Exchange	Optuna	90.25	0.081	**0.169**	**0.280**	**0.681**	15.054	**400**
	Random		**0.080**	0.189	0.391	0.959	**12.106**	2033
ili	Optuna	54	1.506	**1.891**	**2.302**	**3.080**	**7.503**	**400**
	Random		**1.495**	2.363	2.842	4.027	7.532	3976
traffic	Optuna	93	0.387	**0.438**	**0.491**	**0.612**	**1.051**	**400**
	Random		**0.379**	0.487	0.589	0.686	1.473	1018
weather	Optuna	74	0.144	**0.193**	**0.245**	**0.312**	**29.895**	**400**
	Random		**0.143**	0.207	0.263	0.344	109.467	656

We also note that Optuna can provide interpretability. Table J20 shows the importance of each design dimension estimated by Optuna's built-in fANOVA analysis. Sequence Length and Series Normalization contribute the most to performance variation, suggesting their critical role in architecture design.

Table J20: **Relative Importance of Design Dimensions Estimated by Optuna's fANOVA Analysis**

Rank	Design Dimensions	Importance
1	Sequence Length	0.270
2	Series Normalization	0.255
3	Series Embedding	0.134
4	Feature Attention	0.077
5	Series Decomposition	0.053
6	Channel Independent	0.050
7	Series Sampling/Mixing	0.029
8	Epochs	0.025
9	d_model d_ff	0.020
10	Learning Rate	0.020
11	With/Without Timestamps	0.018
12	Network Type	0.017
13	Encoder Layers	0.013
14	Learning Rate Strategy	0.012
15	Loss Function	0.010
16	Series Attention	0.000

Table K21: **Dataset Characteristics Derived from TFB[49].**

dataset	freq	dim	length	stationary	shifting	correlation
ETTm1	mins	7	57600	9.73E-05	-0.06298	0.612
ETTm2	mins	7	57600	0.003043	-0.40555	0.504
ETTh1	hourly	7	14400	0.0012	-0.06136	0.63
ETTh2	hourly	7	14400	0.021788	-0.40381	0.509
ECL	hourly	321	26304	0.00515	-0.07494	0.802
traffic	hourly	862	17544	3.71E-08	0.066992	0.814
weather	mins	21	52696	1.04E-08	0.213569	0.694
Exchange	daily	8	7588	0.359774	0.325341	0.565
ili	weekly	7	966	0.169155	0.721088	0.674
czelan	mins	11	19934	0.159633	-0.15823	0.683
AQShunyi	hourly	11	35064	0.000302	0.018716	0.613
AQWan	hourly	11	35064	0.000275	-0.01141	0.624

K Deeper Insights into Design Choices

To delve deeper into how each component behaves, TSGym formulate 7 most contentious claims in the reaserch community and clarify them with this benchmark. Beyond the 9 datasets already reported in our paper, we added 3 new ones—AQShunyi, AQWan, and CzeLan—selected by jointly considering dataset profiles and computational cost, and every component underwent at least 2,000 experimental runs. Following the dataset-characterization framework proposed in TFB [49], we describe each dataset in terms of dimensionality, sample size, stationarity, distributional drift, and channel correlation, as shown in Table K21.

Claim 1: Model scale should match data scale.

Conclusion 1: Yes. We validate this claim from 2 aspects: 1) Hidden-dimension alignment: On the two high-dimensional datasets (ECL and Traffic), the 256 hidden dimensions surpasses 64 across all four prediction horizons, while for the remaining datasets, 64 outperforms 256 in 32 of 40 settings. 2) Model Backbone: Transformer dominates on the three largest datasets (ECL, Traffic, Weather) across 11 of 12 settings, whereas MLP prevails in 29 of 36 smaller-scale settings.

Claim 2: Transformers are less robust than MLPs.

Conclusion 2: Yes. Using the inter-quartile range (IQR) as the robustness metric, Transformers fare worse than MLPs in 34 out of 48 settings. The average relative IQR gap, $\frac{1}{N}\sum_{i=1}^{N}\frac{IQR_{Transformer,i}-IQR_{MLP,i}}{IQR_{MLP,i}}$, reaches as high as 54.6%.

Claim 3: Transformers exhibit a higher upper bound than MLPs.

Conclusion 3: No. Even when incorporating multiple attention variants, Transformers do not guarantee a performance advantage over MLPs. We compared the upper bound defined as the best result achieved across all of their respective settings. Transformers outperform in 27 of the 48 settings, while underperforming in 21, with an average margin of only +0.038%.

Claim 4: The superiority of Channel-Independent (CI) vs. Channel-Dependent (CD) correlates with inter-channel correlation.

Conclusion 4: No. In the 13 settings where CD models outperform CI models, the average inter-channel correlation is 0.696, versus 0.624 in the 35 settings where CI is preferred. Yet ECL, with the second-highest correlation 0.802, still favors CI on every horizon. This finding suggests the CI/CD decision must consider the data's semantic context, not just statistical correlation.

Claim 5: Series normalization mitigates distribution shift.

Conclusion 5: Yes. On the 6datasets with the highest drift (ETTm2, ETTh2, weather, Exchange, ILI, CzeLan), enabling series normalization wins in all 24 settings. Conversely, on the two near-drift-free datasets (AQShunyi and AQWan), disabling it is better in 8/8 settings. Thus, series normalization effectively combats distribution shift but can discard useful information when no shift is present.

Claim 6: Novel sequence encodings outperform the traditional single-timestep encoding.

Conclusion 6: Yes. Across all 48 settings, classic positional encoding is best only twice, whereas inverted encoding and series patching lead in 17 and 20 settings, respectively.

Claim 7: Novel attention mechanisms outperform vanilla self-attention.

Conclusion 7: Yes. Across 48 settings, vanilla self-attention and auto-correlation each win only 4 times, whereas de-stationary, frequency-enhanced, and sparse attention dominate with 14, 14, and 12 best results, respectively. Notably, on the two least-stationary datasets (Traffic and Weather), De-stationary attention clearly surpasses all other attention and non-attention variants.